

Algoritmos y Estructuras de Datos

Mario Daniel Albarracín
Ingeniero en Electrónica
Universidad Tecnológica Nacional
Profesor Titular de Estructuras de Datos I
Universidad Argentina John F. Kennedy

*del Profesor, Alumno y Asistente
con todo su amor.*

Algoritmos y Estructuras de Datos *Mario Daniel Albaracín*

Es propiedad - Queda hecho el depósito que marca la ley.

Editor: Mario Daniel Albaracín
Ninguna parte del texto de este libro queda ser fotocopiada o reproducida por cualquier medio,
sin la expresa autorización del autor.

Diseño de tapa: Claudio Albaracín
impreso en Argentina

Contenido

Prólogo	13
---------------	----

Capítulo 1. Conceptos Básicos de Algoritmos

1.1	Introducción	15
1.2	Relación entre algoritmo y programa	15
1.3	La computadora como herramienta de trabajo	16
1.4	Diseño de soluciones	16
1.4.1	El análisis del problema	17
1.4.2	La programación	18
1.5	Metodología de programación	19
1.6	Objetos de un programa	19
1.7	Tipos de datos	20
1.8	Constantes	21
1.9	Variables	21
1.10	Operadores	23
1.10.1	Operadores aritméticos	23
1.10.2	Operadores alfanuméricos	24
1.10.3	Operadores relacionales	24
1.10.4	Operadores lógicos	25
1.11	Expresiones	26
1.12	Estructura general de un programa	27
1.13	Instrucciones de entrada	29
1.14	Instrucciones de salida	29
1.15	Instrucciones de asignación	31
1.16	Estructura secuencial	31
Resumen		36
Ejercicios propuestos		38

Capítulo 2. Estructuras de decisión

2.1	Introducción	41
2.2	Instrucciones de decisión	41
2.2.1	Instrucción de decisión simple: SI...ENTONCES	42
2.2.2	Instrucción de decisión doble: SI...ENTONCES...SINO	42
2.2.3	Instrucción de decisión en bloques	43
2.2.4	Instrucción de decisión múltiple: SELECCIONAR CASO	50
Resumen		54
Ejercicios propuestos		56

Capítulo 3. Estructuras de repetición

3.1	Introducción	57
3.2	Instrucciones controladas por condición	57
3.2.1	Instrucción HACER MIENTRAS	57
3.2.2	Instrucción HACER HASTA	58
3.2.3	Ejercicios resueltos	59
3.3	Instrucciones controladas por contador	60
3.3.1	Instrucción PARA...PROXIMO	60
3.3.2	Ejercicios resueltos	61
3.4	Utilización de variables como acumulador	63
3.4.1	Ejercicios resueltos	63
3.5	Utilización de variables como contador de eventos	65
3.5.1	Ejercicios resueltos	65
3.6	Máximos y mínimos	67
	Resumen	72
	Ejercicios propuestos	72

Capítulo 4. Ejercicios de propósito general

4.1	Introducción	77
4.2	Ejercicios resueltos	77
	Resumen	91
	Ejercicios propuestos	93

Capítulo 5. Vectores

5.1	Introducción	99
5.2	Concepto de vector	100
5.3	Inicialización de un vector	102
5.4	Carga de un vector	102
5.5	Ejercicios de aplicación	102
5.6	Máximos y mínimos de un vector	107
5.7	Ordenamiento de vectores	109
5.8	Método del burbujeo	110
	Resumen	115
	Ejercicios propuestos	120

Capítulo 6. Subprogramas

6.1	Introducción	123
6.2	Ejercicios de aplicación	125
	Resumen	140
	Ejercicios propuestos	142

Capítulo 7. Matrices

7.1	Introducción	151
7.2	Carga y lectura del contenido de una matriz	153
7.3	Máximos y mínimos	155
7.4	Ordenamiento de matrices	156
7.5	Ejercicios de aplicación	157
	Resumen	167
	Ejercicios propuestos	170

Capítulo 8. Ejercicios combinados

8.1	Introducción	173
8.2	Ejercicios resueltos	173
8.2.1	Ejercicios con carga con ciclo PARA...PROXIMO	173
8.2.2	Ejercicios con carga con ciclo HACER...MIENTRAS	184
8.2.3	Ejercicios con dos matrices	190
	Ejercicios propuestos	203

Apéndice 1. Resumen de instrucciones	213
--------------------------------------	-----

Bibliografía	219
--------------	-----

Prólogo

El objetivo principal de este libro es introducir al estudiante en la teoría general de los principios de programación a partir de la construcción, desarrollo y aplicación de algoritmos y estructuras de datos internos.

Se intentó reflejar los avances acumulados en las últimas décadas, en los modelos y técnicas utilizados para desarrollar algoritmos, en nuestro caso tomando como herramienta de programación un pseudocódigo de características introductorias, fundamentalmente desde una perspectiva didáctica.

De la misma manera que en trabajos realizados anteriormente, nuevamente se ha puesto especial preocupación en el aspecto pedagógico en cuanto al desarrollo y encadenamiento de los distintos temas tratados y en especial en el aumento del grado de complejidad de los ejercicios, producto de la experiencia acumulada en tantos años de dictado de la materia.

Solo el proceso de construcción del conocimiento por parte del estudiante, permitirá incorporar los conceptos presentados, a partir de los temas tratados, del análisis de los ejercicios resueltos y del desarrollo de los ejercicios propuestos al final de cada capítulo.

A modo de ayuda para facilitar las actividades de estudio, al finalizar cada unidad se incorporó un resumen de los conceptos principales.

El texto está organizado en 8 capítulos y abarca hasta estructuras de datos internas.

Se decidió no incluir estructuras de datos externas (Archivos), por considerar que el grado de abstracción que presenta su estudio teórico, sólo es propio acompañado con la práctica en la computadora y utilizando un lenguaje de programación, lo que no está dentro de los objetivos de este curso sino en una etapa posterior.

El primer capítulo introduce al lector en los conceptos básicos de la programación, por medio de la presentación de los principios y de las definiciones generales.

En los capítulos dos y tres analizan los distintos tipos de estructuras de decisión y repetición respectivamente, con ejemplos de diferentes formas de aplicación para cada caso.

El capítulo cuatro plantea distintos casos de problemas de propósito general.

rel en donde se aplican las herramientas de programación aprendidas en capítulos anteriores.

En el quinto y en el sexto capítulo se estudia el primer tipo de estructura interna de datos: el vector, analizando algoritmos clásicos como los procesos de búsqueda de máximos y mínimos, o el ordenamiento de un conjunto de datos.

Dentro del capítulo siete se presentan las matrices, segundo tipo de estructura interna de datos, con ejercicios resueltos de los procesos más comunes de aplicación.

Para finalizar en el último capítulo, el ocho, se analizan una gran cantidad de ejercicios que combinan la utilización de vectores y matrices.

Agradecemos la posibilidad recibida por parte de la Universidad Argentina John F. Kennedy, al permitirme ejercer la docencia en sus aulas y dedicar día a día.

Al Licenciado Jorge Bonapace y al Licenciado Roberto Coscia por todo el apoyo brindado permanentemente a mi tarea. A Sarita Cuellar y Arnaldo Farías por su colaboración de siempre.

Por último, un agradecimiento especial a mis queridos alumnos.

El autor

Conceptos Básicos de Algoritmos

1.1 INTRODUCCIÓN

Podemos definir como algoritmo al conjunto finito y ordenado de pasos que nos permiten resolver un determinado problema.

Un algoritmo es un método de resolución, son el que podemos resolver sin ambigüedad el problema planteado, a partir de la secuencia de pasos elementales a aplicar.

Cuando hacemos una llamada telefónica, ponemos en marcha un conjunto ordenado y finito de pasos: tomar el teléfono, verificar el tono, etc.

Lo mismo ocurre cuando debemos reemplazar un neumático averiado. Ponemos en marcha un algoritmo: colocar el freno de mano, tomar la rueda de auxilio, tomar las herramientas, etc.

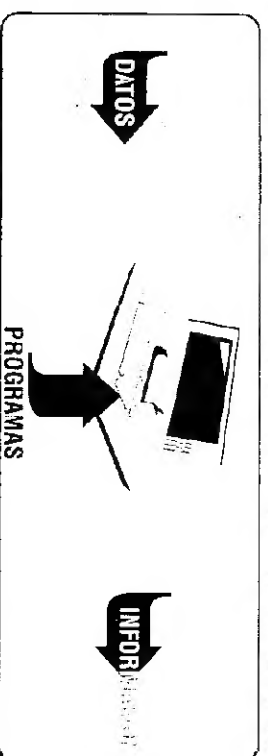
Pasamos gran parte de nuestro tiempo resolviendo problemas que se nos plantean a diario, es así que sin darnos cuenta vivimos desarrollando algoritmos.

1.2 RELACIÓN ENTRE ALGORITMO Y PROGRAMA

Si bien nuestro estudio está orientado a la resolución de problemas, específicamente nos centramos en el tipo de problema a resolver por una computadora. Se trata de proveer a la máquina del algoritmo necesario para que pueda llevar adelante los trabajos planteados. Se deben dar a la máquina las instrucciones que conforman el algoritmo.

Programar es escribir el algoritmo en un lenguaje común para el programador y la máquina.

Se denomina programa al conjunto finito y ordenado de instrucciones dadas a la computadora, para que esta procese los datos a partir de los cuales se obtendrá la información buscada.



Se puede observar en la figura 1, la diferencia conceptual entre dato e información.

Dato es el valor con el que contamos y que constituye una información no elaborada.

El resultado de proceso de los datos a partir de las instrucciones dadas, es la información elaborada.

1.3 LA COMPUTADORA COMO HERRAMIENTA DE TRABAJO

La computadora no es más que una herramienta de trabajo que nos permite tratar información en forma automática, constituyéndose de esta manera en una ayudante eficiente y veloz.

Podemos concluir en función de lo expresado, que el objetivo principal que tenemos que lograr es: que para un problema dado podamos diseñar una solución que pueda ser realizada por la computadora.

1.4 DISEÑO DE SOLUCIONES

Si bien diseñar una solución entra en el terreno de la creatividad y por lo tanto no existen cosas soluciones iguales, podemos distinguir al menos dos etapas a cumplir:

- El análisis del problema.
- La programación propiamente dicha.

10. ALGORITMOS Y ESTRUCTURAS DE DATOS

1.4.1 El análisis del problema

No se puede resolver un problema sin entender completamente el mismo.

¿Qué significa entender un problema?

Si bien no es fácil encontrar la respuesta en nuestra vida, no es tan complicado si ubicamos el problema en el entorno de una máquina. Es así, que diferenciaremos al menos tres partes:

- Identificar los datos: saber con que contamos para a resolución.
- Determinar los resultados: tener claridad para interpretar que tenemos que resolver, que información queremos obtener.
- Explicar las operaciones: encontrar la relación operacional entre los datos y los resultados a obtener.

En el siguiente ejemplo podemos identificar las tres partes:

Determinar el sueldo de un empleado, si el mismo varía de acuerdo a las horas trabajadas durante el mes.

- Datos: las horas trabajadas HT y el valor de la hora VH.
- Resultado: sueldo SDO.
- Relación operacional: $SDO = HT \times VH$.

Cualquier problema puede tener diferentes formas de solución lo que implica que existen distintos tipos de algoritmos con ventajas y desventajas.

¿Cómo elegir el más adecuado?

Al menos distinguimos una serie de características a cumplir:

- Finito: el algoritmo debe finalizar, no puede tener un número de pasos ilimitado.

10. ALGORITMOS Y ESTRUCTURAS DE DATOS

- Legible: debe ser fácil de leer y entender.
- Modificable: debe permitir su permanente actualización y modificación sin dificultad.
- Eficiente: la eficiencia debe comprender tres aspectos
 - Rapidez: bajo tiempo de ejecución.
 - Precisión: fiabilidad de los cálculos realizados.
 - Aprovechamiento de la memoria al máximo.
- Modular: el problema debe dividirse en subproblemas o problemas más pequeños, de manera que la resolución de todos los subproblemas, determine la resolución del problema principal.

1.4.2 La Programación

Hacíamos exp esado anteriormente que programar es explicitar el algoritmo en la computadora.

¿Cómo damos las instrucciones a la máquina?

El problema está en que por ser la computadora una máquina limitada en velocidad, son pocas las operaciones básicas que pueda llevar adelante:

- Operaciones aritméticas sencillas: suma, resta, multiplicación y división.
- Operaciones lógicas sencillas: comparar dos valores y determinar si son iguales o cuál de ellos es el mayor o el menor.
- Almacenar y recuperar información.

La idea es poder escribir instrucciones a partir de estas operaciones sencillas de tal forma que dando un orden lógico a dichas instrucciones, generemos el programa que le permita a la computadora, resolver el trabajo que le

que encomendado.

Durante el desarrollo del curso nos valdremos de un pseudocódigo o pseudolenguaje para escribir nuestros algoritmos.

Este pseudocódigo con finalidad didáctica, cumple en líneas generales con las características de lenguajes estructurados como: BASIC, PASCAL, C, COBOL, etc.

Esto traerá aparejado una rápida adaptación de la teoría a la práctica sobre la computadora.

1.5 METODOLOGÍA DE LA PROGRAMACIÓN

En la resolución de nuestros algoritmos utilizaremos en forma combinada dos métodos: la programación modular y la programación estructurada.

- Programación modular: se basa en la realización de descomposiciones sucesivas del algoritmo inicial que describen el refinamiento progresivo del conjunto de instrucciones que conforman el programa. A este método se lo denomina descendente (TOP DOWN).

• Programación estructurada: el conjunto de instrucciones que constituyen el programa, se agrupan a su vez en subconjuntos denominados estructuras. Existen tres tipos de estructuras que serán abordadas oportunamente:

- estructura secuencial: las instrucciones se ejecutan en el orden que se han codificado.
- estructura de decisión: existen condicionales que de acuerdo a su cumplimiento o no, provocan la ejecución de fases diferentes del programa.
- estructura repetitiva: la ejecución de un grupo de instrucciones se repite un número determinado de veces.

1.6 OBJETOS DE UN PROGRAMA

Se denominan objetos de un programa a todos aquellos manipulados por

las instrucciones. Se pueden distinguir en un objeto tres atributos: nombre, tipo y valor.

Se denomina identificador a las palabras creadas por el programador para dar nombre a los objetos.

1.7 TIPOS DE DATOS

Los datos pueden dividirse en tres grandes grupos:

• **Datos numéricos:** números con los cuales podemos realizar operaciones aritméticas. Pueden ser:

1- Enteros: todos los números positivos o negativos.

Ejemplo:

458, -217, -24.

2- Reales: todos los números positivos y negativos incluyendo a los que no son enteros.

Ejemplo:

-25.9, 37.08, 0.204

• **Datos alfanuméricos** que pueden ser de tres tipos:

1- Letras de la A, a la Z.

2- Números (no se pueden realizar operaciones aritméticas).

3- Caracteres especiales como guiones, paréntesis, etc

• **Datos booleanos:** sólo pueden tener dos valores, verdadero y falso.

- no pueden leerse como datos.

- se forman a partir de los operadores especiales que analizaremos más adelante.

1.8 CONSTANTES

Son objetos cuyo valor permanece invariable a lo largo de la ejecución de un programa.

Las constantes pueden ser números enteros, números reales o caracteres alfanuméricos.

Ejemplo:

50	constante entera.
3.14159	constante real.
"Hola"	constante alfanumérica.

1.9 VARIABLES

Se denominan variables, aquellos objetos cuyo valor puede ser modificado a lo largo de la ejecución del programa.

En otras palabras, una variable no es otra cosa que una zona de la memoria de la computadora referenciada por un nombre, donde se puede almacenar el valor de un dato, que puede cambiar cuando lo deseamos.

El programador elige el nombre de la variable, el cual es aconsejable que no sea muy extenso y que tenga que ver con la información que contiene.

Una variable no es un dato, sino un área de la memoria que contiene al dato.

Para que una variable esté perfectamente definida, debe especificarse:

- Nombre.
- Tipo de dato que almacena.
- Valor inicial.

Como norma escribiremos nombres de variables de no más de ocho caracteres y en letra minúscula.

Cuando escribimos los nombres, obligatoriamente deben comenzar con una letra y no puede tener espacios en blanco. El resto de los caracteres pueden ser números y algunos caracteres especiales.

Ejemplo

```
suelo
promedio
suelo1
suelo2
vol_max
c
```

Es importante remarcar que los lenguajes tienen palabras reservadas que no pueden ser utilizadas como nombres de variables.

Ejemplo

```
IF
THEN
ELSE
WHILE
FOR
NEXT
PRINT
```

Según el tipo de datos que almacenan las variables pueden ser numéricas o alfanuméricas.

Como variables diferentes colocamos el símbolo \$ como último carácter en una variable alfanumérica.

Ejemplo

Variable	Variable numérica
a	variable alfanumérica
a\$	variable alfanumérica
nombre\$	variable numérica
suelo	variable numérica

1.10 OPERADORES

Los operadores representan el tipo de operación a realizar entre dos o más objetos que conforman una instrucción.

Los operadores pueden ser aritméticos, relacionales, lógicos y otros.

1.10.1 Operadores aritméticos

Permiten tratar números de manera aritmética.

La tabla a continuación describe los distintos tipos de operadores, su significado y el orden de prioridad.

Signo	Operación
+	suma
-	resta
*	multiplicación
/	división
^	potenciación

Ejemplo

Expresión	Resultado
2^2	4
4*8	32
16/4	4
32+21	53
53*4	212

1.10.2 Operadores alfanuméricos

Se utilizan para unir datos alfanuméricos. En realidad es un solo tipo de operador que realiza la operación de concatenación.

Signo	Operación
+	concatenación

Ejemplo

Expresión	Resultado
"Hola" + "Pepe"	Hola Pepe
"3" + "," + "1416"	3.1416

Como se puede apreciar, concatenar equivale a unir los datos alfanuméricos

1.10.3 Operadores relacionales

Estos operadores nos permiten comparar dos valores e indicar, si la relación planteada es verdadera o falsa

Signo	Operación
>	mayor
<	menor
==	igual
>=	mayor o igual
<=	menor o igual
<>	distinto

Ejemplo:

Expresión	Resultado
30 > 10	verdadero
15 = 18	falso
17 <> 20	verdadero

1.10.4 Operadores lógicos

Combinan varios operadores relacionales con las reglas del álgebra de Boole, produciendo un valor final de la expresión, que será verdadero o falso.

Signo	Operación
OR	suma lógica
AND	producto lógico
NOT	negación

- OR: expresión que forma es verdadera cuando alguna de las relaciones es verdadera
- AND: la expresión que forma es verdadera cuando todas las relaciones planteadas son verdaderas.
- NOT: invierte el valor final

Ejemplo

Expresión	Resultado
$9 > 4 \text{ OR } 6 = 8$	verdadero
$10 < 20 \text{ AND } 15 > 25$	falso
$\text{NOT } 10 > 3$	falso

1.11 EXPRESIONES

Una expresión es la representación de un cálculo necesario para la obtención de un resultado. Las expresiones pueden ser:

- Numéricas, producen resultados numéricos. Se construyen con operadores aritméticos.
- Alfanuméricas, producen resultados alfanuméricos. Se construyen con operadores alfanuméricos.
- Condicionales, producen resultados verdadero o falso. Se construyen con operadores relacionales y lógicos.

Cuando se realiza una expresión, la prioridad de ejecución de los operadores para determinar el orden en que se efectúan las operaciones, está conforme a la siguiente tabla:

Prioridad	Operación	Prioridad	Operación
1	potenciación	5	relacionales
2	producto y división	6	negación
3	suma y resta	7	producto lógico
4	concatenación	8	suma lógica

Es importante tener en cuenta las siguientes consideraciones:

- Si se desea cambiar el orden de prioridad, se deben utilizar paréntesis.
- En el caso de paréntesis anidados, se comienza siempre por los internos.
- Cuando dos operaciones tienen el mismo nivel de prioridad, la ejecución es de izquierda a derecha.

Tomemos por ejemplo la siguiente expresión:

$$4 * ((18 + 7) * 23 - 35) / 3 \wedge 2 + 5 * 9$$

De acuerdo al nivel de prioridad se realiza de la siguiente forma:

1	$18 + 7 = 25$
2	$25 * 23 = 575$
3	$575 - 35 = 540$
4	$3 \wedge 2 = 9$
5	$14 * 540 = 7560$
6	$7560 / 9 = 840$
7	$3 * 9 = 27$
8	$840 + 27 = 867$

1.12 ESTRUCTURA GENERAL DE UN PROGRAMA

Tal como se mencionó anteriormente, nos valdremos del pseudocódigo para escribir nuestros algoritmos. Utilizaremos un conjunto de palabras las cuales estarán sujetas a las reglas que implican que nuestras instrucciones deben ser interpretadas por el computador.

Debemos tener presente que nuestros algoritmos van a ser escritos ad hoc para poder programar la computadora, a que impere restricciones que debamos respetar.

El pseudocódigo nos permite en forma dídctica, hacer una primera aproximación a cualquier lenguaje real de programación pues sigue las mismas reglas de escritura. Esto significa que se utiliza la metodología estructurada y modular, convirtiéndose en nuestra herramienta de diseño de algoritmos.

En general, un programa escrito en un lenguaje de programación que ha de procesar la computadora, con el objeto de obtener como resultado información a partir de un conjunto de datos de entrada.

Desde un punto de vista funcional, un programa se estructura en tres partes:

- **Entrada de datos:** está formada por todas las instrucciones que toman los datos objeto del programa desde un dispositivo externo (teclado, unidad de disco, etc), depositándolos en la memoria principal de la computadora, incluyendo la depuración o validación de los mismos.

- **Proceso:** es el conjunto de instrucciones que resuelven el problema a partir de los datos que han sido introducidos, dejando los resultados en la memoria principal. El dispositivo es encargado de llevar a cabo esta tarea es la unidad central de proceso o CPU.

- **Salida de resultados:** le constituyen las instrucciones que hacen que los datos resultantes del proceso sean proporcionados al exterior por medio de algún dispositivo (monitor, impresora, etc).

De acuerdo a este tipo, podemos agrupar las instrucciones de la siguiente manera:

- Instrucciones de entrada: permitir ingresar los datos.
- Instrucciones de solución: implementar los cálculos.
- Instrucciones de salida: son operaciones de asignación de valores a variables.

En resumen con nuestro pseudocódigo podemos manipular objetos en general: datos, variables, expresiones, etc.

1.13 INSTRUCCIONES DE ENTRADA

Tomar datos del exterior almacenándolos en variables.

Sintaxis: **INGRESAR variable**

Ejemplo: **INGRESAR numero**

En el ejemplo ingresamos desde el teclado un dato numérico que se almacenará en la variable de nombre "numero".

Si se desean leer varios datos, se pueden colocar en instrucciones consecutivas o bien en la misma instrucción separando las variables con coma.

Sintaxis: **INGRESAR variable1,variable2,...,variableN**

Ejemplo: **INGRESAR alumno\$, nota**

El ingreso de datos puede ser accionado por comandos de activación que estarán delimitados por comillas.

Sintaxis: **INGRESAR "comentario", variable**

Ejemplo: **INGRESAR "ingrese el nombre del alumno:", alumno\$**

1.14 INSTRUCCIONES DE SALIDA

Presentar en pantalla o en impresora comandos, constantes, contenidos de variables y resultados de expresiones.

- Impresión de comentarios

Sintaxis: **IMPRIMIR "comentario"**

Ejemplo: **IMPRIMIR "Buenas Noches"**

Los comentarios son constantes alfanuméricas.

• Impresión de variables

Sintaxis:	IMPRIMIR variable
Ejemplo:	IMPRIMIR sueldo

Se puede mostrar a partir de una única instrucción imprimir, el contenido de más de una variable

Sintaxis:	IMPRIMIR variable1, variable2, ..., variableN
Ejemplo:	IMPRIMIR alumno\$, nota

• Impresión de constantes

Sintaxis:	IMPRIMIR constante
Ejemplo:	IMPRIMIR 57

• Impresión de expresiones

Sintaxis:	IMPRIMIR expresión
Ejemplo:	IMPRIMIR suma / cantidad * 100

Al guardar en el ingreso de datos se pueden combinar en la salida comentarios con variables, etc

Sintaxis:	IMPRIMIR "comentario", variable
Ejemplo:	IMPRIMIR "El sueldo del empleado es:", sueldo

1.15 INSTRUCCIONES DE ASIGNACIÓN:

Almacenan en una variable el valor de una constante, o resultado de una expresión aritmética o bien el contenido de otra variable.

El cambio de valor de una variable se hace mediante la asignación, utilizando como operador el símbolo = que no expresa en este caso una igualdad el valor a la derecha del símbolo se asigna a la variable de la izquierda

Sintaxis:	variable = constante
Ejemplo 1:	cantidad = 3
Ejemplo 2:	nombre\$ = "María Florencia"
Sintaxis:	variable = expresión
Ejemplo 3:	promedio = (nota1 + nota2 + nota3) / 3
Sintaxis:	variable1 = variable2
Ejemplo 4:	a = b

Se podría tener una instrucción de este tipo $c = c + 1$, pues como se dijo la asignación no es una igualdad, a la variable c se le asigna el valor que contiene incrementado en uno

1.16 ESTRUCTURA SECUENCIAL

La estructura secuencial está formada por un conjunto de instrucciones, que se ejecutan una a continuación de la otra conformando una secuencia.

En realidad todo programa es una gran estructura secuencial donde se intercalan estructuras de decisión y repetitiva para controlar el flujo de los datos. En principio nos limitaremos a analizar estructuras secuenciales en su forma más simple, instrucciones de entrada, de asignación y de salida. Los programas que realizaremos, tendrán la siguiente estructura

- ENTRADA
- PROCESO
- SALIDA

Todos los ejercicios deben estar encabezados por una instrucción de arranque: COMIENZO, y finalizar con una orden de terminación: FIN.

A continuación desarrollaremos un conjunto de ejercicios que nos permitirán comprender mejor este tipo de estructura.

Ejercicio 1

Dados dos números cualesquiera sumarlos

```
COMIENZO
INGRESAR "Ingrese dos números ", a, b
suma ← a + b
IMPRIMIR "La suma es ", suma
FIN
```

En este primer ejercicio podemos ver las tres instrucciones mencionadas anteriormente. Se ingresan los datos con INGRESAR, se asignan a la variable suma y se muestra el resultado con la instrucción IMPRIMIR.

Como en todos los ejercicios que analizaremos podemos apreciar como está delimitado el programa por COMIENZO y FIN.

Ejercicio 2

Dados tres números, sumarlos de a dos

```
COMIENZO
INGRESAR "Ingrese tres números ", a, b, c
suma1 ← a + b
suma2 ← suma1 + c
IMPRIMIR "La suma es: ", suma2
FIN
```

Observamos como dos asignaciones componen las instrucciones de proceso.

Ejercicio 3

Dados los tres lados de un triángulo, hallar el perímetro.

```
COMIENZO
INGRESAR "Ingrese el primer lado del triángulo: ", a
INGRESAR "Ingrese el segundo lado del triángulo: ", b
INGRESAR "Ingrese el tercer lado del triángulo: ", c
perimet ← a + b + c
IMPRIMIR "El perímetro es: ", perimet
FIN
```

A diferencia de ejercicio anterior, ahora se ingresan las variables con tres instrucciones INGRESAR. Cualquiera de los formatos mencionados es correcta, será la característica del ejercicio la que determinará cual variante usar.

Ejercicio 4

Dados la base y la altura de un triángulo, hallar el área.

```
COMIENZO
INGRESAR "Ingrese el valor en cm. de la base ", b
INGRESAR "Ingrese el valor en cm. de la altura ", a
area ← b * a / 2
IMPRIMIR "El área del triángulo es: ", area "cm"
FIN
```

Podemos apreciar en la impresión, como a variable numérica area, está ubicada entre dos cadenas de caracteres: "El área de triángulo" y "cm".

Ejercicio 5

Dados los lados de un rectángulo, calcular el perímetro, la superficie y la diagonal.

COMIENZO

INGRESAR "ingrese o valor en cm. de lado mayor", para

INGRESAR "ingrese o valor en cm. del lado menor", para

perim = (may + men) * 2

su = may * men

di = (may ^ 2 + men ^ 2) ^ (1 / 2)

MPRIMIR "El perímetro es:", perim, "cm"

MPRIMIR "La superficie es:", su, "cm"

MPRIMIR "La diagonal es:", diag, "cm"

FIN

Por primera vez se utiliza el símbolo \wedge , para representar la potenciación como operación aritmética. Por otra parte tendremos tres instrucciones: MPR para la salida de resultados que representan informaciones sintéticas.

Ejercicio 6

Dado un valor en kilómetros, expresarlo en metros

COMIENZO

INGRESAR "ingrese un valor en kilómetros", km

mb = 1000 * km

MPR MIR "El valor en metros es:", mb

FIN

Nuevamente nos encontramos con una estructura secuencial muy sencilla donde podemos definir cada una de las instrucciones de manera detallada los, asignación de un cálculo y salida de resultados.

Ejercicio 7

Calcular el sueldo de un empleado, conociendo la cantidad de horas trabajadas. El valor de la hora es de 5 \$

COMIENZO

INGRESAR "ingrese la cantidad de horas trabajadas", horas

sueldo = horas * 5

MPR MIR "El sueldo es:", sueldo, "pesos"

FIN

Este ejercicio responde a la misma estructura que el anterior.

Ejercicio 8

Calcular el sueldo de un empleado, conociendo la cantidad de horas trabajadas y el valor que gana por hora

COMIENZO

INGRESAR "ingrese la cantidad de horas trabajadas", horas

INGRESAR "ingrese el valor de la hora", valor

sueldo = horas * valor

MPRIMIR "El sueldo es:", sueldo, "pesos"

FIN

Leemos una resolución muy similar a la del ejercicio 7, con la salvedad que ahora el valor de las horas es un dato a ingresar desde afuera de programa.

Ejercicio 9

Una compañía de venta de heladeras paga por mes a sus vendedores, 500\$, más un 10% de comisión sobre el total de las ventas que efectúan, conocemos para el vendedor, la cantidad de heladeras vendidas y el importe de la heladera (se comercializa sólo un modelo) ¿Cuál será el sueldo del vendedor?

COMIENZO

INGRESAR "ingrese la cantidad de heladeras vendidas", cantidad

INGRESAR "ingrese el importe de la heladera", importe

sueldo = 500 + cantidad * importe + 0.1

MPRIMIR "El sueldo es:", sueldo, "pesos"

FIN

Por primera vez, calculamos un porcentaje, en este caso del 10%. Dicho valor está expresado por el 0.1 que multiplicamos al importe y cantidad dentro del cálculo del sueldo.

RESUMEN

Algoritmo: secuencia ordenada de pasos que permite la resolución de un problema

Programa: explicación del algoritmo en una computadora

Dato: es toda información sin elaborar que utiliza la computadora

Información: es el resultado del proceso de los datos

Etapas en el diseño de soluciones

- análisis del problema
- Se pueden diferenciar tres partes:

- 1-identificar datos
- 2-determinar resultados
- 3-estructurar las operaciones que vinculen los datos con los resultados

Se pueden distinguir una serie de características a cumplir en la elaboración de un problema: finito, legible, modificable, eficiente y modular

- un algoritmo debe cumplir con:
- las operaciones básicas que puede llevar a cabo una computadora
- 1-operaciones aritméticas sencillas
- 2-operaciones lógicas sencillas
- 3-recorrer y almacenar información

Pseudocódigo: descripción de un algoritmo utilizando una combinación

de palabras y símbolos de programación

Conceptos Básicos de Algoritmos

En las instrucciones escritas en lenguaje común se utilizan lenguaje de programación.

Programación modular: descomponer un problema en subproblemas más fáciles de resolver

Programación estructurada: todo programa debe seguir todo según

- estructura de control
- secuencia de instrucciones se ejecutan en el orden que se indican en el código.

- decisiones de acuerdo al cumplimiento o no de una condición se ejecutan grupos de instrucciones diferentes

- repetición: la ejecución de un grupo de instrucciones se repite un número determinado de veces

Objetos: todos aquellos manipulados por las instrucciones

Atributos: un objeto posee como atributos el nombre, el tipo y el valor

Tipos de datos: numéricos, alfanuméricos y lógicos

Variables: un área de memoria de la computadora que contiene un dato.

Tipo de variables: numéricas, alfanuméricas y lógicas

Constante: valor que permanece invariable a lo largo del proceso

Operadores: se utilizan para manipular datos, pueden ser

- aritméticos
- alfanuméricos
- relacionales
- lógicos
- paréntesis

Algoritmos y Estructuras de Datos

Ejercicios Propuestos

Indicar el valor de x para cada una de las siguientes expresiones

- a) $x = (4 + 6) * 12$
- b) $x = (10 + 22) / 4$
- c) $x = (20 + 7) / 3 ^ 2$
- d) $x = 5 * 5 + 10 * (8 - 6)$

2. Para $a = 5$, $b = 3$ y $c = 2$, indicar el resultado final de las siguientes expresiones

- a) $a * b + c$
- b) $c * b - c ^ 2$
- c) $a * b / (b + c)$
- d) $(a + b + c) * b$

3. Indicar el valor final de a , b , c y d

- a) $c = 3$
 $b = 2$
 $c = a + b$
 $a = c + b$
 $b = c$
- b) $a = 2$
 $b = 6$
 $c = b - a$
 $d = a + b$
 $a = c + 2 * d$
 $b = 5 * d - c ^ 2$
 $c = a * b$
 $d = b + c$

4. Indicar si son variables o falsas las siguientes expresiones

- a) '25' + "25" = 50"
- b) '25' + "25" = '2525'
- c) "espacio" = "espacio"
- c) 'espacio' = "espacio"
- e) 32 + 18 = 50
- f) 53 < 24 AND '2' (6 + 4 + 2)
- g) 24 < 12 OR 9 3 ^ 2
- h) NOT 24 < 12

5. Reduzar un programa que permita hallar la longitud de la hipotenusa de un triángulo. Se tiene como dato las longitudes de los catetos

6. El precio del pasaje para un vuelo es de 680\$ en clase turista y se aplica un incremento de 30% en primera clase. Se desea saber la recaudación obtenida en un vuelo

Estructuras de Decisión

2.1 INTRODUCCIÓN

Controlan el flujo de ejecución al seleccionar que grupo de instrucciones deben ejecutarse. De esta manera mejora el funcionamiento de programa al poder realizarse un mayor número de tareas.

Realizan acciones alternativas, pues la ejecución de una instrucción o grupo de instrucciones depende de si se cumple o no, una o varias condiciones.

2.2 INSTRUCCIONES DE DECISIÓN

Se utiliza la instrucción `SI` y la respuesta puede ser verdadera o falsa, es decir sí o no.

Ejemplo: Si llueve, tomaré un taxi.

La realización de la acción está sujeta a que se cumpla la condición. Existen distintos tipos de formatos para instrucciones de decisión.

- Decisión simple
- Decisión doble
- Decisión en bucles
- Decisión múltiple

2.2.1 Instrucción de decisión simple: SI..... ENTONCES.....

Sintaxis: **SI condición ENTONCES instrucción**
Ejemplo: **SI x > 0 ENTONCES IMPRIMIR x**

La computadora examina la condición. Pueden suceder dos cosas:

- Si se cumple la condición, se ejecuta la instrucción que está a la derecha de ENTONCES y continúa ejecutando las que están fuera del SI.

- Si no se cumple, no entra en el SI y ejecuta las instrucciones siguientes.

Se puede poner más de una condición, siempre y cuando estén unidas por los operadores lógicos OR, AND y NOT.

Sintaxis

SI condición1 operador lógico condición2 ENTONCES instrucción

2.2.2 Instrucción de decisión doble: SI..... ENTONCES..... SI NO.....

Es muy común tener que evaluar dos procesos completamente distintos, dependiendo de si se cumple o no la condición.

Ejemplo: Si llueve, tomaré un taxi, sino, iré caminando.

Traduciendo este tipo de estructura a pseudocódigo, tenemos la siguiente sintaxis: SI..... ENTONCES..... SI NO.....

Sintaxis: **SI condición ENTONCES instrucción1 SI NO instrucción2**
Ejemplo: **SI a >= 0 ENTONCES IMPRIMIR a SI NO IMPRIMIR b**

Es decir:

- Si se cumple la condición se ejecuta la instrucción de imprimir el contenido de la variable a.

- Si no se cumple la condición, se ejecuta el contenido de la variable b.

Veamos con algunos ejemplos:

Ejercicio 1

Dados dos números distintos, imprimir el mayor.

COMIENZO
 INGRESAR "Ingrese los números ", num1, num2
 SI num1 > num2 ENTONCES IMPRIMIR num1 SI NO IMPRIMIR num2
 FIN

En el ejemplo se comparan los dos variables "num1" y "num2", la inversión de la mayor esta condición se realiza la ejecución de la instrucción SI ENTONCES SI NO.

Ejercicio 2

Dado un número decir si es cíclico.

COMIENZO
 INGRESAR "Ingrese un número ", num
 SI num > 0 ENTONCES IMPRIMIR "cíclico" SI NO IMPRIMIR "no es cíclico"
 FIN

Este es un ejemplo de un "if" anterior, con la diferencia que se condice ahora la impresión de un mensaje como se utilizó el "if".

2.2.3 Instrucción de decisión en bloques:

Se utiliza cuando las acciones siguientes a ENTONCES o SI NO están compuestas por varias instrucciones.

Estructuras de Decisión

Sintaxis: Si condición -ENTONCES

instrucción(es)

SINO

instrucción(es)

FIN SI

Ejemplo.

INGRESAR a,b,c

SI $a < 0$ ENTONCES

suma = $a + b + c$

IMPRIMIR "La suma es ", suma

SINO

producto = $a * b * c$

IMPRIMIR "El producto es ", producto

FIN SI

SINO es opcional por lo que la estructura puede escribirse.

Sintaxis Si condición ENTONCES

instrucción(es)

FIN SI

Ejemplo

INGRESAR a

SI $a > b$ ENTONCES

suma = $a + 10$

IMPRIMIR "La suma es ", suma

FIN SI

Los siguientes ejercicios resuélvenlos nos darán una mayor comprensión sobre la utilización de este tipo de estructura.

Ejercicio 3

Dado un número, decir si es positivo, negativo o cero.

44- algoritmos y estructuras de decisión

Estructuras de Decisión

COMIENZO

INGRESAR "Ingrese un número ", num

SI num > 0 ENTONCES

IMPRIMIR "número positivo"

SINO

SI num < 0 ENTONCES

IMPRIMIR "número negativo"

SINO IMPRIMIR "número igual a cero"

FIN SI

FIN SI

FIN

Se puede apreciar el andamiaje de las estructuras SI ENTONCES SINO: una estructura dentro de otra.

Ejercicio 4

Dados dos números, calcular

- la suma, si el primero es mayor que el segundo
- la diferencia, si el primero es menor que el segundo
- El producto, si son iguales.

COMIENZO

INGRESAR "Ingrese dos números.", num1 num2

SI num1 $>$ num2 ENTONCES

suma = num1 + num2

IMPRIMIR "La suma es ", suma

SINO

SI num2 $>$ num1 ENTONCES

resta = num2 - num1

IMPRIMIR "La resta es ", resta

SINO

prod = num1 * num2

IMPRIMIR "El producto es ", prod

FIN SI

FIN SI

FIN

44- algoritmos y estructuras de decisión - 100

Instrucciones de Decisión

Nuevamente al igual que el ejercicio anterior, se darán dos opciones para poder evaluar las condiciones de verdad para el resultado del problema.

Ejercicio 5

Dados tres lados de un triángulo, decir que tipo de triángulo es.

```
COMIENZO
INGRESAR "Ingreso la longitud de los lados ", a, b, c
SI a > b AND b > c ENTONCES
    IMPRIMIR "Triángulo equilátero"
SINO
    SI a < b AND b < c > c ENTONCES
        IMPRIMIR "Triángulo escaleno"
    SINO
        IMPRIMIR "Triángulo isósceles"
FIN SI
```

En este ejercicio se recurre al operador lógico AND, para validar la capacidad del de pregunta en las condiciones evaluadas por la instrucción de decisión.

Así, al colocar a b AND b < c puede hacer una doble pregunta sin tener que escribir dos instrucciones de decisión.

Ejercicio 6

Dados dos números X y Z, sumarlos si X es mayor que Z y restarlos en caso contrario.

```
COMIENZO
INGRESAR "Ingreso dos números: ", x, z
SI x > Z ENTONCES
    res = x + z
SINO
    res = x - z
```

Instrucciones de Decisión II

```
SI x < 7 ENTONCES res = x
FIN SI
IMPRIMIR "Resultado es: ", res
FIN
```

Este ejercicio permite apreciar, características comunes en el planteo de la estructura "ingreso de los datos, evaluación de las condiciones e impresión de resultados".

Ejercicio 7

Dados tres números, se pide:

- La suma
- El promedio
- Si el promedio es mayor que 5, imprimir mensaje que diga "El promedio es mayor que 5"

```
COMIENZO
INGRESAR "Ingreso tres números: ", a, b, c
prom = (a + b + c) / 3
IMPRIMIR "La suma es: ", suma
IMPRIMIR "El promedio es: ", prom
SI prom > 5 ENTONCES IMPRIMIR "El promedio es mayor que 5"
FIN
```

La instrucción de decisión se utiliza en este caso, para imprimir un mensaje que estará condicionado por el contenido de una de las variables numéricas calculadas (prom).

Como variante de resolución se podría haber impreso directamente el cálculo sobre la instrucción de impresión. Observemos como quedaría:

```
COMIENZO
INGRESAR "Ingreso tres números: ", a, b, c
IMPRIMIR "La suma es: ", a + b + c
IMPRIMIR "El promedio es: ", suma / 3
SI prom > 5 ENTONCES IMPRIMIR "El promedio es mayor que 5"
FIN
```

Ejercicio 8

Dados 4 números, decir si la suma de los dos primeros es mayor a la suma de los dos segundos

COMIENZO

INGRESAR "Ingresar cuatro números" a b c d
SI (a + b > (c + d) ENTONCES

IMPRIMIR "La suma de los dos primeros números es mayor"

SINO

IMPRIMIR "La suma de los dos primeros números no es mayor"

FIN SI

FIN

Se puede ver como en lugar de evaluar el contenido de las variables, se comparan dos expresiones: a + b con c + d

Ejercicio 9

Se conocen las edades y estaturas de 3 alumnos de un curso. Se pide:

- Imprimir la edad promedio
- Imprimir la estatura promedio
- Imprimir las edades de los alumnos mayores de 15 años, que no den

mayores de 15 años

COMIENZO

INGRESAR "Ingresar las 3 edades" edad1, edad2, edad3

INGRESAR "Ingresar las 3 estaturas" altura1, altura2, altura3

calcular edad promedio = (edad1 + edad2 + edad3) / 3

calcular altura promedio = (altura1 + altura2 + altura3) / 3

IMPRIMIR "La edad promedio es:" edadprom

IMPRIMIR "La altura promedio es:" alturaprom

SI edad > 15 AND altura < 1.5 ENTONCES IMPRIMIR edad1

SI edad2 > 15 AND altura2 < 1.5 ENTONCES IMPRIMIR edad2

SI edad3 > 15 AND altura3 < 1.5 ENTONCES IMPRIMIR edad3

FIN

Nuevamente al igual que en el ejercicio 5, se vuelve a utilizar el operador lógico AND, esta vez para condicionar doblemente la impresión de los resultados

Ejercicio 10

Una empresa paga sueldos calculando el valor de la hora y la cantidad de horas que trabaja cada empleado. Además, si el empleado trabaja más de 100 horas, le premian con 100\$ y si trabajó más de 200 horas, le dan 50\$ más.

Hallar el sueldo del empleado.

COMIENZO

INGRESAR "Ingresar el valor de la hora:" vh

INGRESAR "Ingresar la cantidad de horas trabajadas:" ht

basico = vh * ht

SI ht > 200 ENTONCES

sueldo = basico + 150

SINO

SI ht > 100 ENTONCES

sueldo = basico + 100

SINO

sueldo = basico

FIN SI

FIN SI

FIN

En este caso se ingresaron dos variables, vh y ht, cuyo producto se asigna a una nueva variable basico. Esta, más la suma del valor adicional según corresponda, se asignará ahora a la variable sueldo para su posterior impresión

Ejercicio 11

Dadas la cantidad de horas trabajadas, la categoría y la antigüedad de un empleado, calcular el sueldo teniendo en cuenta que cobra 50\$ adicionales por cada año trabajado

El valor de la hora para cada categoría es

Instrucciones de Decisión

Categoría 1: 10\$
Categoría 2: 15\$
Categoría 3: 20\$

COMIENZO

INGRESAR "Horas trabajadas", h1
INGRESAR "Categoría", cat
INGRESAR "Antigüedad", ant
SI cat = 1 ENTONCES
sueldo = h1 * 10 + 50 * ant

SINO

SI cat = 2 ENTONCES
sueldo = h1 * 15 + 50 * ant
SINO
sueldo = h1 * 20 + 50 * ant

FIN

Estamos en presencia nuevamente de decisiones anidadas, esto es así porque se evalúan los valores de cat que condiciona el cálculo de los sueldos.

2.2.4 Instrucción de decisión múltiple: SELECCIONAR CASO.....

Esta instrucción permite a seleccionar de un conjunto de acciones a partir de una lista de diferentes opciones o casos. La instrucción de decisión múltiple facilita la selección de más de dos alternativas, haciendo los programas más flexibles y más fáciles de leer y escribir.

Sintaxis

SELECCIONAR CASO variable
CASO condición1
instrucción(es)
CASO condición2
instrucción(es)
CASO condición3

Instrucciones de Decisión

instrucciones)

OTRO CASO
instrucción(es)
FIN SELECCIONAR

SELECCIONAR CASO proporciona el resultado que permite la ejecución de un bloque de sentencias CASO, pues cada secuencia de instrucciones evalúa diferentes resultados de la variable.
La alternativa "opcione" OTRO CASO proporciona un bloque de instrucciones a ejecutarse, sólo en el caso que todos los resultados, evaluados sean falsos.

Ejemplo

SELECCIONAR CASO numero
CASO < 0
IMPRIMIR "Número negativo"
CASO > 0
IMPRIMIR "Número positivo"
OTRO CASO
IMPRIMIR "Número igual a cero"
FIN SELECCIONAR

Se puede apreciar la ventaja que presenta la decisión múltiple en el análisis del siguiente caso.

Si volvemos al ejercicio 11 pero agregando dos categorías más, observaremos como se dificulta la escritura del programa ofreciendo su legibilidad, pues deberíamos colocar el primer dos decisiones múltiples más.

Se produce un entramado de las decisiones, lo cual en el caso de ser nuevas las instrucciones de decisión "anidadas" es poco práctico.

Ahora, en lugar de estar utilizando una decisión múltiple veríamos como la resolución de ejercicio, gana en claridad.

A continuación resolveremos el ejercicio 12 de las dos maneras sin y con decisión múltiple.

Ejercicio 12

Dadas a caridad de horas trabajadas, la categoría y la antigüedad de un empleado, calcular el sueldo teniendo en cuenta que cobra 50\$ adicionales.

Estructuras de Decisión

les por cada año trabajado
El valor de la hora para cada categoría es

Categoría 1 10\$
Categoría 2 12\$
Categoría 3 15\$
Categoría 4 18\$
Categoría 5 20\$

- Caso a resolución 'anidando' S.. ENTONCES SINO

COMIENZO

INGRESAR 'Horas trabajadas:', ht

INGRESAR "Categoría:", cat

INGRESAR "Aptitud:", apt

SI cat = 1 ENTONCES

suelo = ht * 10 + 50 * apt

SINO

SI cat = 2 ENTONCES

suelo = ht * 12 + 50 * apt

SINO

SI cat = 3 ENTONCES

suelo = ht * 15 + 50 * apt

SINO

SI cat = 4 ENTONCES

suelo = ht * 18 + 50 * apt

SINO

suelo = ht * 20 + 50 * apt

FIN SI

FIN SI

FIN SI

FIN

- Caso b resolución utilizando SELECCIONAR CASO

COMIENZO

INGRESAR "Horas trabajadas:", ht

INGRESAR "Categoría:", cat

INGRESAR "Aptitud:", apt

SELECCIONAR CASO cat

Estructuras de Decisión

CASO = 1
vh = 10
CASO = 2
vh = 12
CASO = 3
vh = 15
CASO = 4
vh = 18
CASO = 5
vh = 20

FIN SELECCIONAR

suelo = ht * vh

MPR MIR "El sueldo es:", sueldo

- N

RESUMEN

Las instrucciones de decisión, realicen acciones alternativas, pues la ejecución de una instrucción o grupo de instrucciones depende de si se cumple o no, una o varias condiciones.
Existen distintos tipos de formato para instructores de decisión.

- **Instrucción de decisión simple: SI.....ENTONCES.....**

Sintaxis SI condición ENTONCES instrucción

- **Instrucción de decisión doble: SI.....ENTONCES.....SINO.....**

Sintaxis SI condición ENTONCES instrucción1 SINO instrucción2

- **Instrucción de decisión en bloques:**

Sintaxis SI condición ENTONCES

instrucciones)

SINO

instrucción(es)

FIN SI

SINO es opcional por lo que la instrucción puede escribirse

Sintaxis SI condición ENTONCES

instrucción(es)

FIN SI

- **Instrucción de decisión múltiple: SELECCIONAR CASO.....**

Sintaxis

SELECCIONAR CASO variable

CASO condición1

instrucción(es)

CASO condición2

instrucción(es)

CASO condición3

instrucción(es)

...

...

OTRO CASO

instrucción(es)

FIN SELECCIONAR



Ejercicios Propuestos



- Dados dos números a y b , sumarlos si $a \geq b$ y restarlos si $a < b$.
- Dados dos números a y b , calcular el promedio a/b . Considerar que si b es cero, debe aparecer un mensaje indicando que no puede operarse con b no definido.
- Dados dos números, imprimirlos ordenados de mayor a menor.
- Se conocen las notas de historia y geografía de 3 alumnos de un curso. Se pide:
 - Imprimir el promedio de notas de historia.
 - Imprimir el promedio de notas de geografía.
 - Imprimir el promedio total de cada alumno.

- Dadas la cantidad de horas trabajadas y la categoría de un empleado, calcular el sueldo de bolsillo teniendo en cuenta que los descuentos totales ser del 20%.

El valor de la hora para cada categoría es:

- Categoría 1: 12\$
 Categoría 2: 15\$
 Categoría 3: 18\$
 Categoría 4: 20\$
 Categoría 5: 25\$
 Categoría 6: 28\$
 Categoría 7: 30\$

Estructuras de Repetición

3.1 INTRODUCCIÓN

Este tipo de estructuras nos dan la posibilidad de repetir un conjunto de instrucciones un número determinado de veces.

De acuerdo a la forma de control de la cantidad de repeticiones podemos diferenciar dos grandes grupos de instrucciones:

- controladas por condición.
- controladas por contador.

En el transcurso de este capítulo analizaremos las distintas variantes referidas a este tipo de estructuras. Además introduciremos conceptos de controlaciones eminentemente prácticas como los referidos a la utilización de contadores y acumuladores.

Sobre el final presentaremos el concepto de máximos y mínimos, a partir de algunas aplicaciones.

3.2 INSTRUCCIONES CONTROLADAS POR CONDICIÓN:

3.2.1 Instrucción HACER MIENTRAS

La computadora examina la condición, si se cumple ejecuta las instrucciones que están dentro del ciclo hasta que deje de cumplirse. Entonces se decide si se continúa ejecutando las instrucciones que están a continuación. Si no se cumple la condición no entra en el ciclo.

Sintaxis: HACER MIENTRAS condición
 instrucciones
 REPETIR

Ejemplo:

```
INGRESAR a
HACER MIENTRAS a > 0
    IMPRIMIR a
    INGRESAR a
REPETIR
```

3.2.2 Instrucción HACER HASTA

Es similar a HACER MIENTRAS pero evalúa en forma inversa la condición de tal manera que ejecuta las instrucciones hasta que deje de cumplirse

Sintaxis:

```
HACER
    instrucciones
REPETIR HASTA condición
```

Ejemplo

```
INGRESAR a
HACER
    IMPRIMIR a
    INGRESAR a
REPETIR HASTA a < 0
```

Se puede apreciar en el ejemplo que la estructura HACER HASTA, asegura que por lo menos una vez se va a ejecutar el bloque de instrucciones dentro del ciclo, y que podría no suceder en una estructura HACER MIENTRAS

3.2.3 Ejercicios Resueltos

Para comprender mejor el funcionamiento de las instrucciones de repetición, controladas por condición, analizaremos los siguientes ejercicios.

Ejercicio 1

Dados un conjunto de números, imprimir los números que sean mayores o iguales a 5, suponiendo que se ingresan números hasta uno igual a cero

Algoritmo y estructuras de datos

COMENZAR

```
INGRESAR "Ingrese un número:", num
HACER MIENTRAS num <= 0
```

```
Si num >= 5 ENTONCES IMPRIMIR num
    INGRESAR "Ingrese un número:", num
```

REPETIR

FIN

En el ejercicio 1 se observa como se debe colocar dos veces la instrucción INGRESAR, la primera para poder ingresar por primera vez, en el ciclo a partir de la evaluación de la variable num y la segunda para volver a entrar luego de cada repetición.

Este concepto se repite en todos los ejercicios que utilicen estructuras repetitivas

La impresión del contenido de la variable numérica num, está condicionada por el valor de la misma variable, mayor o igual que 5.

Si resolvemos el mismo ejercicio utilizando la instrucción HACER HASTA, está asegurado que el ciclo se ejecutará al menos una vez, no siendo necesario repetir dos veces el ingreso en la estructura

COMENZAR

```
HACER
    INGRESAR "Ingrese un número:", num
    Si num >= 5 ENTONCES IMPRIMIR num
    REPETIR HASTA num <= 0
FIN
```

Ejercicio 2

Dadas las notas y nombres de alumnos de un curso, imprimir los nombres de los alumnos, cuya nota sea mayor que 7. El ingreso finaliza con nota igual a cero

COMENZAR

```
INGRESAR "Ingrese el nombre y la nota", nom$, nota
HACER MIENTRAS nota <= 0
    Si nota > 7 ENTONCES IMPRIMIR nom$
    INGRESAR "Ingrese el nombre y la nota", nom$, nota
REPETIR
FIN
```

Algoritmo y estructuras de datos

El ejercicio muy similar al anterior en su estructura la diferencia radica en que ahora la impresión de la variable, alfanumérica en este caso, se encuentra condicionada por la variable numérica utilizando HACER HASTA.

```
COMENZO
HACER
    INGRESAR "ingrese el número y la nota", num$, nota
    SI nota > 7 ENTONCES MPRIMIR num$
REPETIR HASTA nota <= 0
FIN
```

3.3 INSTRUCCIONES CONTROLADAS POR CONTADOR:

3.3.1 Instrucción PARA...PRÓXIMO

Se utiliza cuando conozco exactamente la cantidad de veces que se repite la ejecución del bloque de instrucciones que conforman la estructura repetitiva

```
Sintaxis      PARA variable índice - valor inicial A valor final PASO
Incremento

                instrucción(es)

PROXIMO
```

El bloque de instrucciones comienza a repetirse de acuerdo al contador que está definido por el valor inicial de arranque que toma la variable índice, el valor final de parada y un PASO que indica el incremento tipo que tiene en cada vuelta de repetición dicha variable índice. Cuando el incremento del PASO es uno, su escritura es opcional.

Ejemplo:

```
PARA contador = 1 A 50 PASO 1
    INGRESAR "Número.", numero
    IMPRIMIR "Número ingresado", numero
PROXIMO
```

3.3.2 Ejercicios Resueltos

A continuación analizaremos algunos ejercicios donde podemos apreciar el funcionamiento de la instrucción PARA...PROXIMO

Ejercicio 3

Dadas 25 notas y nombres los números que sean mayores o iguales a 5

```
COMENZO
PARA c = 1 A 25
    INGRESAR "ingrese un número:", num
    SI num >= 5 ENTONCES MPRIMIR num
PROXIMO
FIN
```

En este caso se omitió colocar el paso, por ser igual a uno. En los ejercicios que analizamos a continuación, como norma, no colocaremos el paso cada vez que el incremento sea uno. La variable c, es la encargada de contar el número de repeticiones. La impresión del número está condicionada a que sea mayor o igual a cinco.

Ejercicio 4

Dadas 15 notas y nombres de alumnos de un curso, imprimir los nombres de los alumnos cuya nota sea mayor a 7.

```
COMENZO
PARA c = 1 A 15
    INGRESAR "ingrese nombre y nota", nom$, nota
    SI nota > 7 ENTONCES MPRIMIR nom$
PROXIMO
FIN
```

La estructura de este ejercicio es muy similar al anterior, a la diferencia que ahora la variable numérica nota, condiciona la impresión de la variable alfanumérica nom\$.

Ejercicio 5

Dadas n notas y nombres de alumnos de un curso, imprimir los nombres de los alumnos cuya nota sea mayor que 7

```

COMIENZO
INGRESAR "Ingrese la cantidad de alumnos ", n
PARA c = 1 A n
    INGRESAR "ingrese nombre y nota", nom$, nota$
    SI nota >= 7 ENTONCES IMPRIMIR nom$
PROXIMO
FIN
  
```

El ejercicio 5 es una variante del 4. La cantidad de repeticiones que tendrá el ciclo PARA, no está expresado en el enunciado, sino que debe inferirse en la variable n , desde afuera. Esto otorga un alto grado de flexibilidad en la manipulación de la cantidad de datos a procesar.

Ejercicio 6

Dados 10 números, imprimir, para cada uno si es positivo o negativo.

```

COMIENZO
PARA c = 1 A 10
    INGRESAR "Ingrese un número.", num
    SI num > 0 ENTONCES
        IMPRIMIR "Positivo"
    SI NO
        SI num < 0 ENTONCES IMPRIMIR "Negativo"
    PROXIMO
FIN
  
```

Se observa en el ejercicio 6, al bajar juntas a las estructuras de control decisión y repetición.

3.4 UTILIZACIÓN DE VARIABLES COMO ACUMULADOR

Un acumulador es una variable que se incrementa o decrementa en un valor variable

Sintaxis: variable acumulador = variable acumulador + ()
variable

Ejemplo: saldo = saldo + valor

La computadora sume el contenido de la variable valor, a la variable saldo y asigna el resultado a la variable saldo de la siguiente.

3.4.1 Ejercicios Resueltos

Veamos algunos ejemplos de utilización de acumuladores

Ejercicio 7

Dados 15 números, imprimir la suma total.

```

COMIENZO
suma = 0
PARA c = 1 A 15
    INGRESAR "Ingrese un número.", num
    suma = suma + num
PROXIMO
IMPRIMIR "La suma total es ", suma
FIN
  
```

En este ejercicio se utilizó como acumulador a la variable suma, a la que previamente se le asignó cero, para evitar arrastrar algún valor aleatorio que pueda tener a almacenado

Es una buena práctica asignar en todos los casos, un valor inicial (inicializar) a las variables que se usen como acumuladores.

Ejercicio 8

Dados números hasta ingresar uno negativo, imprimir la suma total.

```
COMIENZO
suma = 0
INGRESAR "Ingrese un número.", num
HACER MIENTRAS num > 0
    suma = suma + num
    INGRESAR "Ingrese un número.", num
REPEAT
IMPRIMIR "La suma total es:", suma
FIN
```

Este ejercicio presenta a variante con respecto a 7 en la cantidad de valores a procesar a cuales desconocida. Como se sabe que el ingreso final sea con un número igual a cero, se utilizó una instrucción HACER MIENTRAS para controlar el ingreso de los datos a procesar.

Ejercicio 9

Dados n números, imprimir el promedio

```
COMIENZO
INGRESAR "Ingrese la cantidad de números.", n
suma = 0
PARA cada 1 A n
    INGRESAR "Ingrese un número.", num
    suma = suma + num
PROXIMO
prom = suma / n
IMPRIMIR "El promedio es:", prom
FIN
```

En este caso la cantidad de números a procesar es variable pero siempre conocida al momento de comenzar dicho proceso.

Como ahora se puede la impresión del promedio, este se obtiene de dividir el contenido acumulado en suma, por la cantidad de valores procesados (n).

3.5 UTILIZACIÓN DE VARIABLES COMO CONTADOR DE EVENTOS

Un contador es una variable que se incrementa o decrementa en un **valor constante**.

Sintaxis: variable contador = variable contador + (-) constante

Ejemplo: contador = contador + 1

De manera similar a la variable acumulada, el contador debe ser provisto con un valor inicial a partir del cual comenzará a contar.

3.5.1 Ejercicios Resueltos

Ejercicio 10

Dados 10 números, imprimir cuántos son positivos, cuántos son negativos, y cuántos son cero.

```
COMIENZO
pos = 0
neg = 0
PARA cada 1 A 10
    INGRESAR "Ingrese un número.", num
    SI num > 0 ENTONCES
        pos = pos + 1
    SI num < 0 ENTONCES
        neg = neg + 1
    PROXIMO
IMPRIMIR "Cantidad de positivos:", pos
IMPRIMIR "Cantidad de negativos:", neg
IMPRIMIR "Cantidad de ceros:", ceros
FIN
```


Se trabaja con dos contadores, uno para los números positivos y otro para los negativos. En el caso de la cantidad de ceros se podría haber utilizado otro contador, pero en este caso, se optó por calcularlo por diferencia.

Ejercicio 11

Dados los sueldos de empleados, imprimir cuantos ganan más de 2000\$ y cuantos ganan menos de esa suma.

```
COMIENZO
mas ← 0
menos ← 0
INGRESAR "Ingrese la cantidad de sueldos", n
PARA i ← 1 A n
    INGRESAR "Ingrese el sueldo", sdo
    SI sdo > 2000 ENTONCES
        mas ← mas + 1
    SINO
        SI sdo < 2000 ENTONCES menos ← menos + 1
FIN SI
FIN MO
IMPRIMIR "Cantidad de sueldos mayores a 2000$ ", mas
IMPRIMIR "Cantidad de sueldos menores a 2000$ ", menos
FIN
```

A igual que en el ejercicio 10, se utilizaron dos contadores. Esta vez uno para los sueldos mayores a 2000\$ (mas) y otro para los sueldos menores a 2000\$.

Ejercicio 12

Dadas 20 notas de alumnos de un curso, imprimir:

- Cantidad de alumnos aprobados ($3 < \text{nota} < 4$).
- Cantidad de alumnos desaprobados ($\text{nota} < 3$).
- Cantidad de alumnos promocionados ($\text{nota} > 6$).

```
COMIENZO
apro ← 0
desapro ← 0
pro ← 0
PARA i ← 1 A 20
    INGRESAR "Ingrese la nota", nota
    SI nota > 6 ENTONCES
        pro ← pro + 1
    SINO
        SI nota < 4 ENTONCES
            desapro ← desapro + 1
        SINO
            apro ← apro + 1
FIN SI
FIN MO
PROXIMO
IMPRIMIR "Cantidad de promocionados", pro
IMPRIMIR "Cantidad de desaprobados", desapro
IMPRIMIR "Cantidad de aprobados", apro
FIN
```

En este caso se utilizaron tres contadores, as como se vio previamente en los ejercicios en cero. Además de tener una decisión dentro del ciclo de repetición PARA, agregamos una segunda instrucción de decisión en cada ciclo de la repetición.

3.7 MÁXIMOS Y MÍNIMOS

Una de las aplicaciones clásicas en programación es la obtención de los valores máximo y de un valor mínimo entre un conjunto de valores. Las estructuras de decisión y repetición estudiadas nos brindan las herramientas necesarias para el desarrollo de los algoritmos que construyamos. Las respuestas a nuestros problemas son ejercicios que se analizan a continuación, nos permiten visualizar los mecanismos de resolución de todos los ejercicios como resultado de un máximo o un mínimo. En unidades posteriores analizaremos la posibilidad que exista más de un máximo o de un mínimo.

Ejercicio 13

Dados dos temperaturas, imprimir la mayor (considérense que las temperaturas son distintas).

```

COMIENZO
INGRESAR "ingrese una temperatura: ", temp1
INGRESAR "ingrese otra temperatura: ", temp2
SI temp1 > temp2 ENTONCES IMPRIMIR temp1 SINO IMPRIMIR temp2
FIN

```

En el ejemplo se compararon dos temperaturas y se imprimió la mayor

Ejercicio 14

Dados 5 sueldos, imprimir el máximo

```

COMIENZO
max ← 0
PARA i ← 1 A 5
    INGRESAR "ingrese el sueldo: ", sdo
    SI sdo > max ENTONCES max ← sdo
PROXIMO
IMPRIMIR "El sueldo máximo es: ", max, "$"
FIN

```

Se utilizó una variable max, para almacenar el sueldo máximo. Esta variable se inicializó en cero, para garantizar que en la primera pasada por el ciclo PARA, el primer sueldo sea considerado como máximo. Sólo cuando se ingrese en una futura pasada un valor mayor de sueldo, se reasignará nuevamente max.

Al finalizar la ejecución del ciclo PARA, quedará almacenado en max el valor del sueldo máximo.

En realidad se podría haber inicializado max con cualquier valor que asegure ser menor que los valores de sueldo que se van a procesar.

Otra forma sería la siguiente:

```

8.1. Algoritmo y estructuras de datos

```

```

COMIENZO
INGRESAR "ingrese el sueldo: ", sdo
max ← sdo
PARA i ← 2 A 15
    INGRESAR "ingrese el sueldo: ", sdo
    SI sdo > max ENTONCES max ← sdo
PROXIMO
IMPRIMIR "El sueldo máximo es: ", max, "$"
FIN

```

En este caso se ingresa el primer sueldo y se lo asigna directamente a la variable max, luego se comienza con el ciclo PARA a partir del segundo valor de sueldo (2).

Ejercicio 15

Dados 15 sueldos, imprimir el mínimo

```

COMIENZO
min ← 9999999999
PARA i ← 1 A 15
    INGRESAR "ingrese el sueldo: ", sdo
    SI sdo < min ENTONCES min ← sdo
PROXIMO
IMPRIMIR "El sueldo mínimo es: ", min, "$"
FIN

```

Se puede apreciar que el algoritmo es muy similar al del ciclo del máximo. Se inicializó la variable min con un valor mucho más alto, que garantiza que en la primera pasada por el ciclo PARA, se marque en min el primer sueldo. De la misma manera que en la determinación del máximo, se reasignará min cuando se ingrese un valor inferior al que fue previamente almacenado.

Al finalizar el ciclo, min guardará el menor valor de sueldo de los 15 dados.

Otra resolución del ejercicio igualmente correcta sería

```

8.1. Algoritmo y estructuras de datos

```

Estructuras de Repetición

```

COMIENZO
INGRESAR "Ingrese el sueldo.", sdo
"Fin sdo"
PARA i = 1 A 5
    INGRESAR "Ingrese el sueldo.", sdo
    S sdo < max ENTONCES max = sdo
PROXIMO
MIRAR "El sueldo mínimo es:", min, $
FIN
    
```

En este caso se asigna el sueldo al primer sueldo a manera de ciclo PARA, para poder comparar luego dentro del ciclo con los otros valores.

Ejercicio 16

Dados 5 edades y nombres de alumnos de un curso, imprimir el nombre de alumno de edad máxima.

```

COMIENZO
max = 0
PARA i = 1 A 5
    INGRESAR "Ingrese el nombre del alumno", nombre$
    INGRESAR "Ingrese la edad del alumno", edad
    S edad > max ENTONCES
        max = edad
        nomax = nombre$
PROXIMO
FIN SI
IMPRIMIR "El nombre del alumno de mayor edad es: ", nomax$
FIN
    
```

Observamos como se determina el valor máximo en el ejemplo una variable numérica, pero se imprime el nombre asignado a una variable alfanumérica, que corresponde a ese valor máximo.

Ejercicio 17

En una carrera de autos compiten 45 autos. Al finalizar las vueltas de clasificación se tienen los tiempos de cada auto.

Inicio y desarrollo de datos

Estructuras de Selección

Se pide imprimir

- el número de auto que llegó primero
- el número de auto que llegó último

```

COMIENZO
trn = 0
trn = 9999999999
PARA i = 1 A 45
    INGRESAR "Ingrese el número de auto", auto
    INGRESAR "Ingrese el tiempo de auto", tiempo
    S tiempo < max ENTONCES
        "max = tiempo"
        ultimo = auto
    FIN SI
    S tiempo > min ENTONCES
        "min = tiempo"
        primero = auto
    FIN SI
PROXIMO
IMPRIMIR "El auto de menor tiempo es: ", primero
IMPRIMIR "El auto de mayor tiempo es: ", ultimo
FIN
    
```

En este ejercicio se compara a determinar el número de un máximo (variables max y min) con la de un mínimo (variables trn y prn).

Los procesos forman parte de un mismo ciclo PARA.

Inicio y desarrollo de datos

Estructuras de Repetición

RESUMEN

Este tipo de estructuras nos dan la posibilidad de repetir un conjunto de instrucciones un número determinado de veces.

De acuerdo a la forma de control de la cantidad de repeticiones podemos diferenciar dos grandes grupos de instrucciones controladas por condición y controladas por contador.

Instrucciones controladas por condición.

✓ Instrucción HACER MIENTRAS

La computadora ejecutará la condición, si se cumple ejecutará las instrucciones que están dentro del ciclo hasta que deje de cumplirse. Entonces sale del ciclo y continúa ejecutando las instrucciones que están a continuación.

Si no se cumple la condición no entra en el ciclo.

Sintaxis HACER MIENTRAS condición
 instrucción(es)
REPETIR

✓ Instrucción HACER HASTA

Es similar a HACER MIENTRAS pero evalúa en forma inversa la condición de validez que ejecuta las instrucciones hasta que deje de cumplirse.

Sintaxis HACER
 instrucción(es)
REPETIR HASTA condición

La estructura HACER HASTA asegura que por lo menos una vez se va a ejecutar el bloque de instrucciones dentro del ciclo, lo que podría no suceder en una estructura HACER MIENTRAS.

32. Estructuras de Repetición

Estructuras de Repetición

Instrucciones controladas por contador.

✓ Instrucciones PARA...PROXIMO

Se utiliza cuando conozco exactamente la cantidad de veces que se tiene que repetir la ejecución de bloque de instrucciones que conforman la estructura repetitiva.

Sintaxis: PARA variable índice valor inicial A valor final PASO
 incremento instrucción(es)

PROXIMO

El bloque de instrucciones comienza a repetirse de acuerdo al contador que está definido por el valor inicial de arranque que toma la variable índice, el valor final de parada y un PASO que indica el incremento fijo que tiene en cada vuelta de repetición dicha variable índice.

Cuando el incremento del PASO es uno, su escritura es opcional.

✓ Acumulador

Un acumulador es una variable que se incrementa o decrementa en un valor variable.

Sintaxis: variable acumulador = variable acumulador +/- variable

Es una buena práctica asignar en todos los casos, un valor inicial (inicializar) a las variables que se utilizarán como acumuladores, para evitar arrastrar algún valor a priori o que pueda tener almacenado.

✓ Contador

Los contadores son variables que se incrementan o decrecen en un valor constante.

Sintaxis: variable contador variable contador +/- constante

De manera similar a la variable acumulador, el contador debe ser inicializado.

32. Estructuras de Repetición

mente cargarlo con un valor inicial a partir del cual comenzará a correr

✓ Máximos y Mínimos

Una de las aplicaciones clásicas en programación, es la obtención de un valor máximo o de un valor mínimo entre un conjunto de valores. Las estructuras de decisión y repetición estudiadas, nos brindan las herramientas necesarias para el desarrollo de los algoritmos que constituyen la respuesta a nuestro problema.



Ejercicios Propuestos

1. Dados 50 sueldos, muestre:
 - a) la suma
 - b) cantidad de sueldos mayores que 1500\$
2. Dadas las edades y estaturas de 45 alumnos de un curso, se pide:
 - a) edad promedio,
 - b) estatura promedio
 - c) cantidad de alumnos mayores de 10 años
 - d) cantidad de alumnos que midan menos de 1.40 mts.
3. Una empresa elaboradora de dos productos incrementa las ventas diarias de cada uno de ellos. En cada factura de venta se registra la siguiente información:

Número de factura
Número de artículo
Cantidad en litros
Precio unitario por litro

 - a) ingreso de datos finaliza con número de factura igual a cero. Se pide:
 - a) facturación mensual
 - b) ¿Cuántos litros se vendieron del artículo 1?
 - c) ¿Cuántas facturas de más de 300\$ se emitieron?

4. En una Universidad los alumnos tienen una nota que resulta de sacar el promedio de todas las materias. Voy 700 alumnos.

Se pide

- la cantidad de alumnos con nota promedio superior a 6.
- si la cantidad de alumnos con nota promedio inferior a 4 es mayor o menor a 500, imprimir un mensaje que diga "Esta Universidad tiene un promedio muy bueno".
- Dados los sueldos, imprimir el máximo
- Dados los nombres de artículos y sus correspondientes precios, imprimir
- el nombre del artículo más caro.
- el precio del artículo más barato

Ejercicios de Propósito General

4.1 INTRODUCCIÓN

En este capítulo desarrollaremos una serie de ejercicios que resumen todos los conceptos aprendidos hasta ahora.

El objetivo principal es aprender a razonar las soluciones desde el punto de vista algorítmico, utilizando las herramientas descritas en los capítulos anteriores.

Para lograr lo expresado anteriormente, trataremos de combinar los conocimientos adquiridos y así poder integrarlos en la resolución de los ejercicios planteados.

4.2 EJERCICIOS RESUELTOS

Ejercicio 1

En una empresa los empleados cobran un sueldo según la categoría a la que pertenecen. Son 50 empleados.

Los sueldos son

Categoría 1	500\$
Categoría 2	700\$
Categoría 3	2000\$

Al sueldo se le suma además 100\$ por cada año trabajado.

Si para cada empleado se conoce su categoría y antigüedad, se puede calcular e imprimir:

- la cantidad de empleados por categoría.
- El total de sueldos pagados por categoría.
- El sueldo promedio

Ejercicios de Programación General

d) Salario máximo y la categoría a la que pertenece

CCMENZO

ce1 = 0 ce2 = 0 ce3 = 0

ts1 = 0 ts2 = 0 ts3 = 0

smax = 0

PARA empleado = 1 A 50

INGRESAR "Categoría:", cat

INGRESAR "Atigüedad:", ant

SELECCIONAR CASO cat

CASO = 1

sdo = 1500 + ant * 100

ts1 = ts1 + sdo

ce1 = ce1 + 1

CASO = 2

sdo = 1/0C + ant * 100

ts2 = ts2 + sdo

ce2 = ce2 + 1

CASO = 3

sdo = 2000 + ant * 100

ts3 = ts3 + sdo

ce3 = ce3 + 1

FINSELECCIONAR

SI sdo > smax ENTONCES:

smax = sdo

cmax = cat

FIN SI

PROXIMO

spron = (ts1 + ts2 + ts3) / 50

IMPRIMIR "Cantidad de empleados de la categoría 1:", ce1

IMPRIMIR "Cantidad de empleados de la categoría 2:", ce2

IMPRIMIR "Cantidad de empleados de la categoría 3:", ce3

IMPRIMIR "Total de sueldos de la categoría 1:", ts1, "\$"

IMPRIMIR "Total de sueldos de la categoría 2:", ts2, "\$"

IMPRIMIR "Total de sueldos de la categoría 3:", ts3, "\$"

IMPRIMIR "Sueldo promedio:", spron, "\$"

IMPRIMIR "El sueldo máximo es:", smax, "\$ y corresponde a la categoría:", cmax

FIN

Ejercicios de Programación General

Lo primero que debemos hacer cuando comenzamos a escribir un programa, es inicializar todas las variables que vamos a utilizar como contadores, acumuladores, máximos y mínimos.

Para la resolución se utilizaron tres contadores para la cantidad de empleados, ce1, ce2 y ce3, y tres acumuladores de sueldo por categoría, ts1, ts2 y ts3, los que prevalecieron fuerce iniciados a cero. **Por una cuestión práctica con respecto a la escritura, se separaron estas asignaciones iniciales por dos puntos (:), para no tener que cambiar permanentemente de línea de instrucción.** Esto último es válido en la mayoría de los lenguajes de programación.

La variable empleado se utilizó como variable de control del ciclo PARA y se de 1 a 50.

En la entrada de datos se ingresa la categoría y la antigüedad luego se evalúa a partir de una selección múltiple, la categoría para determinar que variables tendrán en cuenta los datos leídos, para el cálculo de los sueldos y la cantidad de empleados por categoría.

A continuación se determinó el valor del sueldo máximo (smax) y de la categoría a la que pertenece (cmax). **Con respecto a la variable cmax, no fue necesario colocarla en cero en el comienzo, pues en realidad no almacena un valor máximo, sino la categoría a la cual pertenece dicho valor.**

La variable spron almacena el resultado de promedio de sueldos y tamaño debe ser previamente inicializada, pues no tiene en cuenta ningún valor anterior.

Por último tenemos la impresión de resultados.

Ejercicio 2

Una empresa conoce para cada uno de los siguientes datos

nombre

sueldo

categoría

Hay 100 empleados distribuidos en tres categorías

Se pide calcular e imprimir:

a) Total de sueldos en pesos, que paga la empresa

b) Cantidad de empleados que ganan más de 2000\$

Ejercicios de Propósito General

- Cantidad de empleados que ganan menos de 500\$ cuya categoría sea 1.
- Nombre del empleado que gana más.
- Sueldo máximo.
- Total de sueldos en pesos de cada categoría
- Sueldo promedio

CONCAT

totaldo = 0; mas2000 = 0; menos500 = 0

cat1 = 0; cat2 = 0; cat3 = 0

smax = 0

PARA empleado = 1 A 100

INGRESAR "Nombre.", nom\$

INGRESAR "Sueldo.", sueldo

SELECCIONAR "Categoría.", cat

SELECCIONAR CASO cat

CASO = 1

cat1 = cat1 + sueldo

Si sueldo < 500 ENTONCES mas2000 = mas2000 + 1

CASO = 2

cat2 = cat2 + sueldo

CASO = 3

cat3 = cat3 + sueldo

FIN SELECCIONAR

totaldo = totaldo + sueldo

Si sueldo > 2000 ENTONCES mas2000 = mas2000 + 1

Si sueldo > smax ENTONCES

smax = sueldo

nomax\$ = nom\$

FIN SI

PROX MO

sprom = totaldo / 100

IMPRIMIR "Total de sueldos que paga la empresa.", totaldo, "\$"

IMPRIMIR "Empleados que ganan más de 2000\$:", mas2000

IMPRIMIR "Empleados de categoría 1 que ganan menos de 500\$:",

menos500

IMPRIMIR "Empleado que gana más:", nomax\$

IMPRIMIR "Sueldo máximo:", smax, "\$"

IMPRIMIR "Total de sueldos de la categoría 1", cat1, "\$"

Ejercicios de Propósito General

IMPRIMIR "Total de sueldos de la categoría 2.", cat2, "\$"

IMPRIMIR "Total de sueldos de la categoría 3.", cat3, "\$"

IMPRIMIR "Sueldo promedio", sprom, "\$"

FIN

Se utilizaron dos contadores: mas2000, menos500, y cuatro acumulados: totaldo, cat1, cat2 y cat3. Además se trabajó con smax para asignar el máximo

De acuerdo al enunciado de ejercicio, el ingreso de datos se realizó dentro de un ciclo PARA de 100 repetidores, una para cada empleado

Con la decisión múltiple SELECCIONAR CASO, se evaluó la variable cat para acumular los sueldos de acuerdo a la categoría. En el caso particular de cat = 1, se incrementó el contador menos500 en los casos que el sueldo era inferior a 500\$.

Con totaldo se acumularon todos los sueldos independientemente de la categoría. Este acumulador se podía haber evitado al guardar directamente la sumas parciales de sueldo por categoría, o almacenadas en cada uno de los acumuladores cat1, cat2 y cat3

El promedio se obtiene de dividir el total acumulado por la cantidad de empleados, asignando el resultado en la variable sprom

Ejercicio 3

Una línea aérea vende pasajes en 3 aeropuertos. En cada uno de ellos hay tres empleados que son los que efectúan las ventas. Cada vez que un cliente compra pasajes, se registran los siguientes datos

- Número de aeropuerto
- Número de empleado
- Valor del pasaje
- Cantidad de pasajes

El ingreso de datos finaliza con un número de aeropuerto igual a cero. Los números de los empleados se denotan del 1 al 9. Cada cliente puede comprar más de un pasaje.

Se pide calcular e imprimir:

- La cantidad de pasajes vendidos por cada empleado.
- La recaudación por aeropuerto.

1. Ejercicios de Proposición General

- El número de empleado que haya vendido mayor cantidad de pasajes en una venta
- La cantidad de pasajes vendidos por aeropuerto
- El porcentaje de ventas en pasaje de cada empleado, sobre el total
- La cantidad de venta, que haya excedido los 5000\$

COMIENZO

$cpe1 = 0$ $cpe2 = 0$ $cpe3 = 0$ $cpe4 = 0$ $cpe5 = 0$ $cpe6 = 0$ $cpe7 = 0$ $cpe8 = 0$ $cpe9 = 0$
 $ra1 = 0$ $ra2 = 0$ $ra3 = 0$
 $re1 = 0$ $re2 = 0$ $re3 = 0$ $re4 = 0$ $re5 = 0$ $re6 = 0$ $re7 = 0$ $re8 = 0$ $re9 = 0$
 $pa1 = 0$ $pa2 = 0$ $pa3 = 0$
 $c500 = 0$ $c7max = 0$

INGRESAR "Número de aeropuerto", ra
 HACER MIENTRAS ra <= 0
 INGRESAR "Número de empleado", ne
 INGRESAR "Valor del pasaje", yp
 INGRESAR "Cantidad de pasajes", cp
 $ya = cp * yp$
 SELECCIONAR CASO re

CASO

CASO - 2

$cpe1 = cpe1 + cp$
 $re1 = re1 + ya$
 $cpe2 = cpe2 + cp$
 $re2 = re2 + ya$

CASO 3

$cpe3 = cpe3 + cp$
 $re3 = re3 + ya$

CASO - 4

$cpe4 = cpe4 + cp$
 $re4 = re4 + ya$

CASO - 5

$cpe5 = cpe5 + cp$
 $re5 = re5 + ya$

CASO - 6

$cpe6 = cpe6 + cp$
 $re6 = re6 + ya$

CASO - 7

$cpe7 = cpe7 + cp$

1. Ejercicios de Proposición General

CASO 8

$re7 = re7 + ya$

CASO - 9

$cpe8 = cpe8 + cp$
 $re8 = re8 + ya$

CASO - 9

$cpe9 = cpe9 + cp$
 $re9 = re9 + ya$

FIN SELECCIONAR

SELECCIONAR CASO ra

CASO - 1

$ra1 = ra1 + ya$
 $pa1 = pa1 + cp$

CASO - 2

$ra2 = ra2 + ya$
 $pa2 = pa2 + cp$

CASO - 3

$ra3 = ra3 + ya$
 $pa3 = pa3 + cp$

FIN SELECCIONAR

SI rep > c7max FIN TONQUEFS

$c7max = c7$
 $c7max = ra$

FIN SI

$S ya > 5000$ ENTONCES $c5000 = 5000 + ya$
 INGRESAR "Número de aeropuerto", ne

REPETIR

$ra1 = ra1 + ra2 + ra3$
 $pa1 = pa1 + pa2 + pa3$
 $pa2 = pa2 + pa3 + pa4$
 $pa3 = pa3 + pa4 + pa5$
 $pa4 = pa4 + pa5 + pa6$
 $pa5 = pa5 + pa6 + pa7$
 $pa6 = pa6 + pa7 + pa8$
 $pa7 = pa7 + pa8 + pa9$
 $pa8 = pa8 + pa9 + pa10$
 $pa9 = pa9 + pa10 + pa11$
 $pa10 = pa10 + pa11 + pa12$
 $pa11 = pa11 + pa12 + pa13$
 $pa12 = pa12 + pa13 + pa14$
 $pa13 = pa13 + pa14 + pa15$
 $pa14 = pa14 + pa15 + pa16$
 $pa15 = pa15 + pa16 + pa17$
 $pa16 = pa16 + pa17 + pa18$
 $pa17 = pa17 + pa18 + pa19$
 $pa18 = pa18 + pa19 + pa20$
 $pa19 = pa19 + pa20 + pa21$
 $pa20 = pa20 + pa21 + pa22$
 $pa21 = pa21 + pa22 + pa23$
 $pa22 = pa22 + pa23 + pa24$
 $pa23 = pa23 + pa24 + pa25$
 $pa24 = pa24 + pa25 + pa26$
 $pa25 = pa25 + pa26 + pa27$
 $pa26 = pa26 + pa27 + pa28$
 $pa27 = pa27 + pa28 + pa29$
 $pa28 = pa28 + pa29 + pa30$
 $pa29 = pa29 + pa30 + pa31$
 $pa30 = pa30 + pa31 + pa32$
 $pa31 = pa31 + pa32 + pa33$
 $pa32 = pa32 + pa33 + pa34$
 $pa33 = pa33 + pa34 + pa35$
 $pa34 = pa34 + pa35 + pa36$
 $pa35 = pa35 + pa36 + pa37$
 $pa36 = pa36 + pa37 + pa38$
 $pa37 = pa37 + pa38 + pa39$
 $pa38 = pa38 + pa39 + pa40$
 $pa39 = pa39 + pa40 + pa41$
 $pa40 = pa40 + pa41 + pa42$
 $pa41 = pa41 + pa42 + pa43$
 $pa42 = pa42 + pa43 + pa44$
 $pa43 = pa43 + pa44 + pa45$
 $pa44 = pa44 + pa45 + pa46$
 $pa45 = pa45 + pa46 + pa47$
 $pa46 = pa46 + pa47 + pa48$
 $pa47 = pa47 + pa48 + pa49$
 $pa48 = pa48 + pa49 + pa50$
 $pa49 = pa49 + pa50 + pa51$
 $pa50 = pa50 + pa51 + pa52$
 $pa51 = pa51 + pa52 + pa53$
 $pa52 = pa52 + pa53 + pa54$
 $pa53 = pa53 + pa54 + pa55$
 $pa54 = pa54 + pa55 + pa56$
 $pa55 = pa55 + pa56 + pa57$
 $pa56 = pa56 + pa57 + pa58$
 $pa57 = pa57 + pa58 + pa59$
 $pa58 = pa58 + pa59 + pa60$
 $pa59 = pa59 + pa60 + pa61$
 $pa60 = pa60 + pa61 + pa62$
 $pa61 = pa61 + pa62 + pa63$
 $pa62 = pa62 + pa63 + pa64$
 $pa63 = pa63 + pa64 + pa65$
 $pa64 = pa64 + pa65 + pa66$
 $pa65 = pa65 + pa66 + pa67$
 $pa66 = pa66 + pa67 + pa68$
 $pa67 = pa67 + pa68 + pa69$
 $pa68 = pa68 + pa69 + pa70$
 $pa69 = pa69 + pa70 + pa71$
 $pa70 = pa70 + pa71 + pa72$
 $pa71 = pa71 + pa72 + pa73$
 $pa72 = pa72 + pa73 + pa74$
 $pa73 = pa73 + pa74 + pa75$
 $pa74 = pa74 + pa75 + pa76$
 $pa75 = pa75 + pa76 + pa77$
 $pa76 = pa76 + pa77 + pa78$
 $pa77 = pa77 + pa78 + pa79$
 $pa78 = pa78 + pa79 + pa80$
 $pa79 = pa79 + pa80 + pa81$
 $pa80 = pa80 + pa81 + pa82$
 $pa81 = pa81 + pa82 + pa83$
 $pa82 = pa82 + pa83 + pa84$
 $pa83 = pa83 + pa84 + pa85$
 $pa84 = pa84 + pa85 + pa86$
 $pa85 = pa85 + pa86 + pa87$
 $pa86 = pa86 + pa87 + pa88$
 $pa87 = pa87 + pa88 + pa89$
 $pa88 = pa88 + pa89 + pa90$
 $pa89 = pa89 + pa90 + pa91$
 $pa90 = pa90 + pa91 + pa92$
 $pa91 = pa91 + pa92 + pa93$
 $pa92 = pa92 + pa93 + pa94$
 $pa93 = pa93 + pa94 + pa95$
 $pa94 = pa94 + pa95 + pa96$
 $pa95 = pa95 + pa96 + pa97$
 $pa96 = pa96 + pa97 + pa98$
 $pa97 = pa97 + pa98 + pa99$
 $pa98 = pa98 + pa99 + pa100$
 $pa99 = pa99 + pa100 + pa101$
 $pa100 = pa100 + pa101 + pa102$
 $pa101 = pa101 + pa102 + pa103$
 $pa102 = pa102 + pa103 + pa104$
 $pa103 = pa103 + pa104 + pa105$
 $pa104 = pa104 + pa105 + pa106$
 $pa105 = pa105 + pa106 + pa107$
 $pa106 = pa106 + pa107 + pa108$
 $pa107 = pa107 + pa108 + pa109$
 $pa108 = pa108 + pa109 + pa110$
 $pa109 = pa109 + pa110 + pa111$
 $pa110 = pa110 + pa111 + pa112$
 $pa111 = pa111 + pa112 + pa113$
 $pa112 = pa112 + pa113 + pa114$
 $pa113 = pa113 + pa114 + pa115$
 $pa114 = pa114 + pa115 + pa116$
 $pa115 = pa115 + pa116 + pa117$
 $pa116 = pa116 + pa117 + pa118$
 $pa117 = pa117 + pa118 + pa119$
 $pa118 = pa118 + pa119 + pa120$
 $pa119 = pa119 + pa120 + pa121$
 $pa120 = pa120 + pa121 + pa122$
 $pa121 = pa121 + pa122 + pa123$
 $pa122 = pa122 + pa123 + pa124$
 $pa123 = pa123 + pa124 + pa125$
 $pa124 = pa124 + pa125 + pa126$
 $pa125 = pa125 + pa126 + pa127$
 $pa126 = pa126 + pa127 + pa128$
 $pa127 = pa127 + pa128 + pa129$
 $pa128 = pa128 + pa129 + pa130$
 $pa129 = pa129 + pa130 + pa131$
 $pa130 = pa130 + pa131 + pa132$
 $pa131 = pa131 + pa132 + pa133$
 $pa132 = pa132 + pa133 + pa134$
 $pa133 = pa133 + pa134 + pa135$
 $pa134 = pa134 + pa135 + pa136$
 $pa135 = pa135 + pa136 + pa137$
 $pa136 = pa136 + pa137 + pa138$
 $pa137 = pa137 + pa138 + pa139$
 $pa138 = pa138 + pa139 + pa140$
 $pa139 = pa139 + pa140 + pa141$
 $pa140 = pa140 + pa141 + pa142$
 $pa141 = pa141 + pa142 + pa143$
 $pa142 = pa142 + pa143 + pa144$
 $pa143 = pa143 + pa144 + pa145$
 $pa144 = pa144 + pa145 + pa146$
 $pa145 = pa145 + pa146 + pa147$
 $pa146 = pa146 + pa147 + pa148$
 $pa147 = pa147 + pa148 + pa149$
 $pa148 = pa148 + pa149 + pa150$
 $pa149 = pa149 + pa150 + pa151$
 $pa150 = pa150 + pa151 + pa152$
 $pa151 = pa151 + pa152 + pa153$
 $pa152 = pa152 + pa153 + pa154$
 $pa153 = pa153 + pa154 + pa155$
 $pa154 = pa154 + pa155 + pa156$
 $pa155 = pa155 + pa156 + pa157$
 $pa156 = pa156 + pa157 + pa158$
 $pa157 = pa157 + pa158 + pa159$
 $pa158 = pa158 + pa159 + pa160$
 $pa159 = pa159 + pa160 + pa161$
 $pa160 = pa160 + pa161 + pa162$
 $pa161 = pa161 + pa162 + pa163$
 $pa162 = pa162 + pa163 + pa164$
 $pa163 = pa163 + pa164 + pa165$
 $pa164 = pa164 + pa165 + pa166$
 $pa165 = pa165 + pa166 + pa167$
 $pa166 = pa166 + pa167 + pa168$
 $pa167 = pa167 + pa168 + pa169$
 $pa168 = pa168 + pa169 + pa170$
 $pa169 = pa169 + pa170 + pa171$
 $pa170 = pa170 + pa171 + pa172$
 $pa171 = pa171 + pa172 + pa173$
 $pa172 = pa172 + pa173 + pa174$
 $pa173 = pa173 + pa174 + pa175$
 $pa174 = pa174 + pa175 + pa176$
 $pa175 = pa175 + pa176 + pa177$
 $pa176 = pa176 + pa177 + pa178$
 $pa177 = pa177 + pa178 + pa179$
 $pa178 = pa178 + pa179 + pa180$
 $pa179 = pa179 + pa180 + pa181$
 $pa180 = pa180 + pa181 + pa182$
 $pa181 = pa181 + pa182 + pa183$
 $pa182 = pa182 + pa183 + pa184$
 $pa183 = pa183 + pa184 + pa185$
 $pa184 = pa184 + pa185 + pa186$
 $pa185 = pa185 + pa186 + pa187$
 $pa186 = pa186 + pa187 + pa188$
 $pa187 = pa187 + pa188 + pa189$
 $pa188 = pa188 + pa189 + pa190$
 $pa189 = pa189 + pa190 + pa191$
 $pa190 = pa190 + pa191 + pa192$
 $pa191 = pa191 + pa192 + pa193$
 $pa192 = pa192 + pa193 + pa194$
 $pa193 = pa193 + pa194 + pa195$
 $pa194 = pa194 + pa195 + pa196$
 $pa195 = pa195 + pa196 + pa197$
 $pa196 = pa196 + pa197 + pa198$
 $pa197 = pa197 + pa198 + pa199$
 $pa198 = pa198 + pa199 + pa200$
 $pa199 = pa199 + pa200 + pa201$
 $pa200 = pa200 + pa201 + pa202$
 $pa201 = pa201 + pa202 + pa203$
 $pa202 = pa202 + pa203 + pa204$
 $pa203 = pa203 + pa204 + pa205$
 $pa204 = pa204 + pa205 + pa206$
 $pa205 = pa205 + pa206 + pa207$
 $pa206 = pa206 + pa207 + pa208$
 $pa207 = pa207 + pa208 + pa209$
 $pa208 = pa208 + pa209 + pa210$
 $pa209 = pa209 + pa210 + pa211$
 $pa210 = pa210 + pa211 + pa212$
 $pa211 = pa211 + pa212 + pa213$
 $pa212 = pa212 + pa213 + pa214$
 $pa213 = pa213 + pa214 + pa215$
 $pa214 = pa214 + pa215 + pa216$
 $pa215 = pa215 + pa216 + pa217$
 $pa216 = pa216 + pa217 + pa218$
 $pa217 = pa217 + pa218 + pa219$
 $pa218 = pa218 + pa219 + pa220$
 $pa219 = pa219 + pa220 + pa221$
 $pa220 = pa220 + pa221 + pa222$
 $pa221 = pa221 + pa222 + pa223$
 $pa222 = pa222 + pa223 + pa224$
 $pa223 = pa223 + pa224 + pa225$
 $pa224 = pa224 + pa225 + pa226$
 $pa225 = pa225 + pa226 + pa227$
 $pa226 = pa226 + pa227 + pa228$
 $pa227 = pa227 + pa228 + pa229$
 $pa228 = pa228 + pa229 + pa230$
 $pa229 = pa229 + pa230 + pa231$
 $pa230 = pa230 + pa231 + pa232$
 $pa231 = pa231 + pa232 + pa233$
 $pa232 = pa232 + pa233 + pa234$
 $pa233 = pa233 + pa234 + pa235$
 $pa234 = pa234 + pa235 + pa236$
 $pa235 = pa235 + pa236 + pa237$
 $pa236 = pa236 + pa237 + pa238$
 $pa237 = pa237 + pa238 + pa239$
 $pa238 = pa238 + pa239 + pa240$
 $pa239 = pa239 + pa240 + pa241$
 $pa240 = pa240 + pa241 + pa242$
 $pa241 = pa241 + pa242 + pa243$
 $pa242 = pa242 + pa243 + pa244$
 $pa243 = pa243 + pa244 + pa245$
 $pa244 = pa244 + pa245 + pa246$
 $pa245 = pa245 + pa246 + pa247$
 $pa246 = pa246 + pa247 + pa248$
 $pa247 = pa247 + pa248 + pa249$
 $pa248 = pa248 + pa249 + pa250$
 $pa249 = pa249 + pa250 + pa251$
 $pa250 = pa250 + pa251 + pa252$
 $pa251 = pa251 + pa252 + pa253$
 $pa252 = pa252 + pa253 + pa254$
 $pa253 = pa253 + pa254 + pa255$
 $pa254 = pa254 + pa255 + pa256$
 $pa255 = pa255 + pa256 + pa257$
 $pa256 = pa256 + pa257 + pa258$
 $pa257 = pa257 + pa258 + pa259$
 $pa258 = pa258 + pa259 + pa260$
 $pa259 = pa259 + pa260 + pa261$
 $pa260 = pa260 + pa261 + pa262$
 $pa261 = pa261 + pa262 + pa263$
 $pa262 = pa262 + pa263 + pa264$
 $pa263 = pa263 + pa264 + pa265$
 $pa264 = pa264 + pa265 + pa266$
 $pa265 = pa265 + pa266 + pa267$
 $pa266 = pa266 + pa267 + pa268$
 $pa267 = pa267 + pa268 + pa269$
 $pa268 = pa268 + pa269 + pa270$
 $pa269 = pa269 + pa270 + pa271$
 $pa270 = pa270 + pa271 + pa272$
 $pa271 = pa271 + pa272 + pa273$
 $pa272 = pa272 + pa273 + pa274$
 $pa273 = pa273 + pa274 + pa275$
 $pa274 = pa274 + pa275 + pa276$
 $pa275 = pa275 + pa276 + pa277$
 $pa276 = pa276 + pa277 + pa278$
 $pa277 = pa277 + pa278 + pa279$
 $pa278 = pa278 + pa279 + pa280$
 $pa279 = pa279 + pa280 + pa281$
 $pa280 = pa280 + pa281 + pa282$
 $pa281 = pa281 + pa282 + pa283$
 $pa282 = pa282 + pa283 + pa284$
 $pa283 = pa283 + pa284 + pa285$
 $pa284 = pa284 + pa285 + pa286$
 $pa285 = pa285 + pa286 + pa287$
 $pa286 = pa286 + pa287 + pa288$
 $pa287 = pa287 + pa288 + pa289$
 $pa288 = pa288 + pa289 + pa290$
 $pa289 = pa289 + pa290 + pa291$
 $pa290 = pa290 + pa291 + pa292$
 $pa291 = pa291 + pa292 + pa293$
 $pa292 = pa292 + pa293 + pa294$
 $pa293 = pa293 + pa294 + pa295$
 $pa294 = pa294 + pa295 + pa296$
 $pa295 = pa295 + pa296 + pa297$
 $pa296 = pa296 + pa297 + pa298$
 $pa297 = pa297 + pa298 + pa299$
 $pa298 = pa298 + pa299 + pa300$
 $pa299 = pa299 + pa300 + pa301$
 $pa300 = pa300 + pa301 + pa302$
 $pa301 = pa301 + pa302 + pa303$
 $pa302 = pa302 + pa303 + pa304$
 $pa303 = pa303 + pa304 + pa305$
 $pa304 = pa304 + pa305 + pa306$
 $pa305 = pa305 + pa306 + pa307$
 $pa306 = pa306 + pa307 + pa308$
 $pa307 = pa307 + pa308 + pa309$
 $pa308 = pa308 + pa309 + pa310$
 $pa309 = pa309 + pa310 + pa311$
 $pa310 = pa310 + pa311 + pa312$
 $pa311 = pa311 + pa312 + pa313$
 $pa312 = pa312 + pa313 + pa314$
 $pa313 = pa313 + pa314 + pa315$
 $pa314 = pa314 + pa315 + pa316$
 $pa315 = pa315 + pa316 + pa317$
 $pa316 = pa316 + pa317 + pa318$
 $pa317 = pa317 + pa318 + pa319$
 $pa318 = pa318 + pa319 + pa320$
 $pa319 = pa319 + pa320 + pa321$
 $pa320 = pa320 + pa321 + pa322$
 $pa321 = pa321 + pa322 + pa323$
 $pa322 = pa322 + pa323 + pa324$
 $pa323 = pa323 + pa324 + pa325$
 $pa324 = pa324 + pa325 + pa326$
 $pa325 = pa325 + pa326 + pa327$
 $pa326 = pa326 + pa327 + pa328$
 $pa327 = pa327 + pa328 + pa329$
 $pa328 = pa328 + pa329 + pa330$
 $pa329 = pa329 + pa330 + pa331$
 $pa330 = pa330 + pa331 + pa332$
 $pa331 = pa331 + pa332 + pa333$
 $pa332 = pa332 + pa333 + pa334$
 $pa333 = pa333 + pa334 + pa335$
 $pa334 = pa334 + pa335 + pa336$
 $pa335 = pa335 + pa336 + pa337$
 $pa336 = pa336 + pa337 + pa338$
 $pa337 = pa337 + pa338 + pa339$
 $pa338 = pa338 + pa339 + pa340$
 $pa339 = pa339 + pa340 + pa341$
 $pa340 = pa340 + pa341 + pa342$
 $pa341 = pa341 + pa342 + pa343$
 $pa342 = pa342 + pa343 + pa344$
 $pa343 = pa343 + pa344 + pa345$
 $pa344 = pa344 + pa345 + pa346$
 $pa345 = pa345 + pa346 + pa347$
 $pa346 = pa346 + pa347 + pa348$
 $pa347 = pa347 + pa348 + pa349$
 $pa348 = pa348 + pa349 + pa350$
 $pa349 = pa349 + pa350 + pa351$
 $pa350 = pa350 + pa351 + pa352$
 $pa351 = pa351 + pa352 + pa353$
 $pa352 = pa352 + pa353 + pa354$
 $pa353 = pa353 + pa354 + pa355$
 $pa354 = pa354 + pa355 + pa356$
 $pa355 = pa355 + pa356 + pa357$
 $pa356 = pa356 + pa357 + pa358$
 $pa357 = pa357 + pa358 + pa359$
 $pa358 = pa358 + pa35$

Ejercicios de Propósito General

COMIENZO

$cpe1 = 0; cpe2 = 0; cpe3 = 0; cpe4 = 0; cpe5 = 0; cpe6 = 0$
 $r1 = 0; r2 = 0; r3 = 0$
 $p1 = 0; p2 = 0; p3 = 0$
 menos200 = 0
 $cpmin = 999999999$
 PAXA ventas = 1 A 1500

INGRESAR "Número de terminal", n1

INGRESAR "Número de empleado", ne

INGRESAR "Valor de pasaje", vp

INGRESAR "Cantidad de pasajes", cp

vía cp * vp

SELECCIONAR CASO re

CASO = 1

reol cpe1 + cp

CASO = 2

reol cpe2 + cp

CASO = 3

reol cpe3 + cp

CASO = 4

reol cpe4 + cp

CASO = 5

reol cpe5 + cp

CASO = 6

reol cpe6 + cp

FIN SELECCIONAR

SELECCIONAR CASO n1

CASO =

r1 = r1 + v1a

p1 = p1 + cp

CASO = 2

r2 = r2 + v1a

p2 = p2 + cp

CASO = 3

r3 = r3 + v1a

p3 = p3 + cp

FIN SELECCIONAR

SELECCIONAR

SI cp < cpmin ENTONCES

cpmin = cp

reol = re

FIN SI

Ejercicios de Propósito General

Si v1a < 200 ENTONCES menos200 = menos200 + 1

PROXIMO

reol p1 + p2 + p3

reol p1 / p1 * 100

reol p2 / p1 * 100

reol p3 / p1 * 100

MPRIM R "Cantidad de pasajes vendidos por el empleado 1", cp1

MPRIM R "Cantidad de pasajes vendidos por el empleado 2", cp2

MPRIM R "Cantidad de pasajes vendidos por el empleado 3", cp3

MPRIM R "Cantidad de pasajes vendidos por el empleado 4", cp4

MPRIM R "Cantidad de pasajes vendidos por el empleado 5", cp5

MPRIM R "Cantidad de pasajes vendidos por el empleado 6", cp6

MPRIM R "Recaudación de la terminal 1", r1, "\$"

MPRIM R "Recaudación de la terminal 2", r2, "\$"

MPRIM R "Recaudación de la terminal 3", r3, "\$"

MPRIM R "El empleado, nemur, fue el que menos pasajes vendió"

MPRIM R "Cantidad de pasajes vendidos en la terminal 1", p1

MPRIM R "Cantidad de pasajes vendidos en la terminal 2", p2

MPRIM R "Cantidad de pasajes vendidos en la terminal 3", p3

MPR M1R "Porcentaje de ventas de la terminal 1", por1, "%"

MPR M1R "Porcentaje de ventas de la terminal 2", por2, "%"

MPR M1R "Porcentaje de ventas de la terminal 3", por3, "%"

MPR M1R "Ventas menores a 200\$", menos200

FIN

Se utilizó en este ejercicio 12 acumuladores: 6 para los pasajes vendidos por cada empleado (cpe) y 6 para los pasajes (p) y la recaudación (r) por terminal. La variable menos200 es el único contador que permite determinar la cantidad de ventas menores a doscientos pesos. La variable nemur almacena el apellido del empleado registrado en cantidad de pasajes por un empleado y funciona en forma paralela a la variable nemur, que almacena el número de empleado que realizó dicha venta. Se utilizó aleatoriamente el valor 999999999 para inicializar cpmin, esta puede ser iniciada con cualquier valor lo suficientemente grande como para asegurar que en una primera pasada por el ciclo, el primer valor leído irá a parar a cpmin. Se trabajó nuevamente con dos decisiones o selecciones múltiples: una para los empleados y otra para las terminales.

Los tres porcentajes pedidos, se calcularon por cantidad de pasaserveridos

Ejercicio 5

Una casa de comidas tiene 6 mesas y 3 mozos. Cada mozo atiende dos mesas y cuando la mesa se desocupa se registran los siguientes datos:

- Número de mozo.
- Número de mesa
- Importe de la cuenta
- Cantidad de personas que comieron.

La cantidad de cuentas a procesar es de 500.
Se pide calcular e imprimir:

- La cantidad de personas atendidas por cada mozo
- La recaudación por mozo
- El número de mozo que haya tenido la cuenta de mayor importe
- La cantidad de personas que comieron por mesa
- El porcentaje de ventas en pesos, de cada mozo sobre el total
- La cantidad de cuentas con importe por encima de los 100\$.

COMIENZO

```
cpmo1 = 0; cpmo2 = 0; cpmo3 = 0
reco1 = 0; reco2 = 0; reco3 = 0
cuente1 = 9999999999
cpme1 = 0; cpme2 = 0; cpme3 = 0; cpme4 = 0; cpme5 = 0; cpme6 = 0
mas100 = 0
PARA ventas = 1 A 500
```

```
INGRESAR "Número de mozo", numozo
INGRESAR "Número de mesa", numesa
INGRESAR "Importe de la cuenta", imp
INGRESAR "Cantidad de personas que comieron", cp
SELECCIONAR CASO numozo
```

CASO - 1

```
cpmo1 = cpmo1 + cp
reco1 = reco1 + imp
```

CASO - 2

```
cpmo2 = cpmo2 + cp
reco2 = reco2 + imp
```

CASO 3

```
cpmo3 = cpmo3 + cp
reco3 = reco3 + imp
```

FIN SELECCIONAR

SELECCIONAR CASO numesa

CASO 1

cpme1 = cuente1 + cp

CASO 2

cpme2 = cpmo2 + cp

CASO 3

cpme3 = cpmo3 + cp

CASO 4

cpme4 = cpmo4 + cp

CASO 5

cpme5 = cpmo5 + cp

CASO 6

cpme6 = cpmo6 + cp

FIN SELECCIONAR

SI imp < cuente1 ENTONCES

cuente1 = imp

mas100 = numozo

- N SI

SI mas100 > 100 ENTONCES mas200 = mas200 + 1

PROXIMO

total = reco1 + reco2 + reco3

por1 = reco1 / total * 100

por2 = reco2 / total * 100

por3 = reco3 / total * 100

IMPRIMIR "Personas atendidas por el mozo 1", cpmo1

IMPRIMIR "Personas atendidas por el mozo 2", cpmo2

IMPRIMIR "Personas atendidas por el mozo 3", cpmo3

IMPRIMIR "Recaudación del mozo 1", reco1, \$

IMPRIMIR "Recaudación del mozo 2", reco2, "\$"

IMPRIMIR "Recaudación del mozo 3", reco3, "\$"

IMPRIMIR "Número de mozo que tuvo la menor cuenta", numozo

IMPRIMIR "Cantidad de personas atendidas en la mesa 1", cpmo1

IMPRIMIR "Cantidad de personas atendidas en la mesa 2", cpmo2

Ejercicios de Propósito General

IMIRIMIR "Cantidad de personas atendidas en la mesa 3" cpre3
 IMIRIMIR "Cantidad de personas atendidas en la mesa 4" cpre4
 IMIRIMIR "Cantidad de personas atendidas en la mesa 5" cpre5
 IMIRIMIR "Cantidad de personas atendidas en la mesa 6" cpre6
 IMIRIMIR "Porcentaje de las ventas del mozo 1" por1,%
 IMIRIMIR "Porcentaje de las ventas del mozo 2" por2,%
 IMIRIMIR "Porcentaje de las ventas del mozo 3" por3,%
 IMIRIMIR "Cantidad de cuentas con importe mayor a 100\$" mas100
 FN

Tomemos en ese caso "2 acumulado es, ó para la cantidad de personas atendidas por mesa (cpme) y ó para la recaudación (ecmo) y la cantidad de personas atendidas (cpamo) por mozo

Se robujó nuevamente con sólo un contador, para la cantidad de cuentas con importe mayor a 100\$

La variable cuenta almacena el importe mínimo para una cuenta y "a baja asociada a mozo n que guarda el número de mozo responsable de la atención de dicha cuenta

Los porcentajes están calculados en función de las recaudaciones por mozo

Ejercicios de Propósito General

RESUMEN

Recordaremos algunos consideraciones prácticas surgidas del análisis de los ejercicios

✓ Lo primero que debemos hacer cuando comenzamos a escribir un programa, es inicializar todas las variables que vamos a utilizar como contadores, acumuladores, máximos y mínimos

✓ Por una cuestión práctica con respecto a la escritura es conveniente separar estas asignaciones, inicializaciones por dos bloques (1), para no tener que cambiar permanentemente líneas de instrucciones. Esto último es válido en la mayoría de los lenguajes de programación

✓ Las variables que tienen asignados máximos o mínimos, no se incrementan o decrementan, pues serían inútiles, pues no tienen en cuenta ningún valor

✓ Las variables que almacenan el resultado de promedios, tampoco deben ser previamente inicializadas, pues no tienen en cuenta ningún valor

✓ El promedio se obtiene de dividir el total acumulado por la cantidad de valores considerados

✓ En los ciclos HACER MIENTRAS se repite las veces que se indica la instrucción de ingreso del valor de la variable que condiciona su acceso, primero para arrancar el ciclo se lo coloca fuera de él luego al finalizar el ciclo, antes de REPETIR, para volver a evaluar la condición de corte.

✓ Los porcentajes se calculan como el cociente entre el valor "dividido" sobre el valor total multiplicado todo por 100

✓ A guisa de las variables utilizadas para almacenar promedios, las

variables a las cuales se les asigna el cálculo de los porcentajes, no necesitan ser inicializadas al comienzo del programa pues no tienen en cuenta valores previos.

✓ Se utilizó aleatoriamente el valor 9999999999 para inicializar una variable, pero, esta puede ser iniciada con cualquier valor lo suficiente grande como para asegurar que en una primera pasada por el ciclo, el primer valor eido ira a parar a la variable que se inicializó con el mínimo



Ejercicios Propuestos



1. Una empresa 'ex' desea procesar sus ventas. Cada vez que una persona realiza una compra se le entrega una factura donde consta:

- Número de factura
- Código del artículo
- Cantidad del artículo
- Precio unitario.

En cada factura se registra un solo código de artículo los artículos son cuatro. El ingreso de datos finaliza con un número de factura igual a cero. Se pide calcular e imprimir:

- a) Total general facturado en pesos
- b) Cantidad de unidades vendida para cada uno de los artículos
- c) Total de artículos vendidos
- d) Cantidad de facturas emitidas para cada uno de los artículos
- e) Número de facturas con mayor importe en pesos
- f) Número de artículos con menor cantidad pedida en una factura.
- g) Porcentaje de ventas en pesos de cada uno de los artículos sobre el total

2. Un supermercado tiene tres secciones

- Almacén
- Verdulería
- Panadería

El supermercado tiene en total seis cajas, dos para cada sección. Cada vez que una persona realiza una compra, se le entrega un recibo donde constan los siguientes datos:

- Número de caja
- Número de sección.
- Importe de la venta
- Cantidad de productos comprados

Ejercicios de Propósito General

Si el importe de la venta supera los 50\$, se le hace un descuento del 10% sobre el total.

Un comprador sin pagar al que se le entrega los centenos queda registrado en la cinta de máquina registradora.

Con dichos datos se desea saber:

- La cantidad de comprobantes emitidos en cada caja.
- La recaudación de cada caja.
- La caja en la que se registró la venta de mayor importe.
- El porcentaje de ventas en pesos de cada caja, sobre el total de las ventas.
- La cantidad de comprobantes con importe mayor a 100\$, que hayan comprado menos de tres productos.
- El sueldo de cada cajero (cada caja o gacha el 10% de la recaudación de su caja).

3. Una cartelería de teatro, vende entradas para distintas obras, en cada una de sus dos sucursales.

En la actualidad, las obras que se ofrecen son cuatro:

- | | |
|--------------------------|------|
| 1. Muñeca de cera | 25\$ |
| 2. El arretre | 15\$ |
| 3. El día de los muertos | 20\$ |
| 4. Los miserables | 30\$ |

Para que el procesamiento de la información sea claro, si una persona compra entradas para dos obras distintas, se le emiten dos comprobantes distintos. Si compra varias entradas para la misma obra, se le entrega un sólo comprobante, donde se indica cuantas compra.

En cada comprobante se registra:

- Número de suceso.
- Número de obra.
- Cantidad de lugares.

5. Compra más de 10 entradas para una misma obra, se le aplica un descuento del 15% sobre el precio total.

Se pide calcular e imprimir:

4. Algoritmo y estructura de datos

Ejercicios de Propósito General

c) La cantidad de entradas vendidas para cada obra.

e) La recaudación por suceso.

d) El número de suceso que haya vendido mayor cantidad de entradas en una venta.

c) La ganancia de la cartelería (le queda el 20% de su recaudación).

e) El porcentaje de ventas, en cantidad de entradas, de cada suceso.

f) La cantidad de entradas vendidas en la sucursal 1 de "Muñeca de cera".

4. Un noticiero de televisión tiene tres comentaristas políticos que cubren alternadamente:

- Casa de gobierno.
- Ministerio de Economía.
- Ministerio de Cultura y Educación.
- Ministerio de Trabajo.

Los comentaristas reciben su paga por horas, siendo el valor de estos:

- | | |
|----------------|------|
| Comentarista 1 | 15\$ |
| Comentarista 2 | 20\$ |
| Comentarista 3 | 25\$ |

Al terminar una jornada de trabajo, cada uno de los comentaristas, entrega en el canal una planilla con los siguientes datos:

- Número de comentarista.
- Número del lugar del trabajo.
- Cantidad de horas trabajadas.
- Cantidad de personas entrevistadas.

5. La cantidad de horas trabajadas en un día es mayor que 10, se le entrega un plus de 40\$.

Se pide calcular e imprimir:

- La cantidad de personas entrevistadas por cada comentarista.
- El sueldo de cada comentarista.
- La cantidad de horas trabajadas en cada lugar de trabajo.
- El número de comentarista con mayor cantidad de horas trabajadas, en un día de trabajo.

5. Algoritmo y estructura de datos

Ejercicios de Propósito General

- El sueldo promedio
- La cantidad de veces que los comentaristas trabajan más de 15 horas en casa de gobierno, o un día de trabajo

5. Una empresa de autos vende pasajes a tres destinos del interior del país

- Córdoba.
- Mendoza
- Tucumán.

Los chicos tienen dos buses y por consiguiente dos tarifas diferentes:

- | | |
|------------|-------|
| 1. Turista | 50\$. |
| 2. Privada | 70\$ |

El costo del pasaje es el mismo para los tres destinos

Al pasajero se le entrega un ticket, donde se consignan los siguientes datos:

- Número de ticket.
- Número de destino.
- Número de clase.
- Cantidad de pasajes

Si una persona compra más de 15 pasajes, se le descuenta un 20% del precio total

Se pide calcular e imprimir:

- La cantidad de pasajes vendidos a cada destino.
 - La recaudación por clase de pasajes
 - El número de tickets con mayor importe, en una venta
 - La cantidad de pasajes vendidos por clase
 - El porcentaje de ventas, en cantidad de pasajes, de cada destino
 - La cantidad de pasajes vendidos a Córdoba en primera clase
6. Un restaurante tiene 6 mesas y 3 mozos. Cada mozo atiende dos mesas y cuando la mesa se desocupa se registra los siguientes datos

08 - algoritmos y estructuras de datos

Ejercicios de Propósito General

- Número de mozo.
- Número de mesa
- Importe de la cuenta.
- Cantidad de personas que comieron

El ingreso de datos finaliza con un número de mozo igual a cero

- La cantidad de personas atendidas por cada mozo
- La recaudación por mozo
- El número de mozo que haya atendido a cuenta de mayor importe
- La cantidad de personas que comieron por mesa
- El porcentaje de ventas en pesos, de cada mozo sobre el total
- La cantidad de cuentas con importe por debajo de los 50\$

09 - algoritmos y estructuras de datos

Vecores

5.1 INTRODUCCIÓN

Hasta el momento, para hacer mención a un dato utilizábamos una variable.

La dificultad se presenta cuando tenemos una gran cantidad de datos que estar relacionados entre sí. Para cada uno de esos datos se debería definir una variable distinta o que ocasionalmente gran dificultad cuando tenemos que escribir un programa, debido a la cantidad de variables a utilizar.

Tomemos como ejemplo el ejercicio 3 desarrollado en el capítulo 4.

“Una línea aérea vende pasajes en 3 aeropuertos. En cada uno de ellos hay tres empleados que son los que efectúan las ventas.”

Cuando calculamos las ventas por empleado utilizamos 9 acumuladores, desde `cpel` hasta `cped`, pues tenemos 3 empleados por cada uno de los 3 aeropuertos.

¿Qué sucedería si en lugar de tener 3 aeropuertos nos hubiéramos referido a 302?

Necesitaríamos 90 acumuladores, o que acumáramos ciudades para escribir las 90 variables `cpel`.

Para resolver estos problemas se agrupan los datos en un mismo conjunto, bajo un nombre común, que se pueden tratar como una sola unidad. A estos conjuntos se los denomina estructuras de datos.

Las estructuras de datos, de acuerdo al lugar donde se almacenan se clasifican en:

✓ **Internas**: se almacenan en la memoria de la computadora, se clasifican en vectores y matrices según el tipo.

✓ **Externas:** se almacenan en soportes externos como los discos, se denominan archivos.

En este capítulo estudiaremos los vectores

5.2 CONCEPTO DE VECTOR

Un vector es un conjunto de datos de mismo tipo, almacenados en la memoria de la computadora.

Cada dato se denomina elemento del vector y está referenciado por una variable **índice**, que indica la posición de dicho elemento dentro del vector.

Previo al uso de un vector, hay que reservar una zona de memoria para su uso, así como definir el número de elementos necesarios para acceder a cada elemento de la estructura. Al proceso de reservar una zona de la memoria para almacenar los datos del vector, se lo denomina **dimensionar** y se realiza de la siguiente forma:

Sintaxis: **Nombre del vector (cantidad de elementos)**

Ejemplo: **contador(20)**

El ejemplo indica que el vector de nombre contador, está compuesto por 20 elementos y por consiguiente puede almacenar 20 datos bajo un mismo nombre

La cantidad de elementos es el máximo valor que puede tomar el índice

Los vectores se dimensionan una sola vez a comienzo del programa para reservar la zona de memoria que se utilizará

Los índices sólo pueden tomar valores enteros. Si el índice sobrepasa la dimensión del vector se producirá un error

La posición del elemento al que queremos acceder, va entre paréntesis. Debemos diferenciar perfectamente los conceptos de posición y elemento. Analicemos el ejemplo de la figura 1:

Posición (índice)	num (5)
1	25
2	32
3	78
4	84
5	93

Figura 1

En la figura uno vemos representado el vector num compuesto por 5 elementos.

```
num(1) = 25
num(2) = 32
num(3) = 78
num(4) = 84
num(5) = 93
```

- La **posición** es la dirección del lugar que ocupa cada dato dentro del vector y está indicado por el índice en el ejemplo serían 1, 2, 3, 4 y 5. El **elemento** es el dato que hay almacenado en el vector, en esta posición. en nuestro ejemplo los datos 25, 32, 78, 84, 93.

Los vectores, al residir en la memoria de la computadora, son de acceso directo, por lo que podemos situarnos directamente en la posición deseada. Los elementos de un vector se utilizan en forma ordenada a la de cualquier variable, interviniendo en instrucciones de

- ✗ asignación variable = vector(i) o vector(i) = vector j
- ✗ errada INPUT\$AK vector(i)
- ✗ salida PRINT#R vector(i)
- ✗ corriendo vector(i) = vector(i) + cantidad
- ✗ acumulador vector(i) = vector(j) + variable

Los vectores que contienen datos alfanuméricos, deben escribirse con el símbolo \$ al final vector\$(i)

5.3 INICIALIZACIÓN DE UN VECTOR

En los vectores siempre colocamos de antemano los límites entre los que variará el índice para realizar su recorrido. Esto es posible porque lo primero que debemos hacer para crear un vector es dimensionarlo. Así, si queremos colocar en cero el vector de nombre contador de 20 elementos lo haríamos de la siguiente manera:

```
contador(20)
PARA i = 1 A 20
    contador(i) = 0
PROXIMO
```

Como podemos apreciar se recurre a un ciclo PARA PROXIMO, para colocar en cero cada uno de los elementos de vector, de esta manera se repetirá 20 veces la asignación.

5.4 CARGA DE UN VECTOR

De la misma manera andada en el punto anterior, recurramos a un ciclo PARA PROXIMO, para cargar un vector durante el ingreso de datos. Consideremos ahora el vector nombres de 10 elementos en el cual vamos a cargar 10 nombres que ingresaremos desde afuera del programa:

```
nombres$(10)
PARA i = 1 A 10
    INGRESAR "ingrese el nombre", nombres$(i)
PROXIMO
```

En este caso, se repetirá 10 veces el ingreso de nombres que se irán cargando en un orden secuencial, en cada uno de las posiciones del vector.

5.5 EJERCICIOS DE APLICACIÓN

Desarrollaremos ahora un conjunto de ejercicios que nos permitan desarrollar un mejor entendimiento del funcionamiento de un vector y de su gran utilidad como herramienta de programación.

PROBLEMAS Y EJERCICIOS DE APLICACIÓN

Ejercicio 1

Cargar un vector de 50 elementos e imprimir:

- El cuarto elemento
- El segundo elemento
- Los elementos en orden invertido
- El producto entre el primer elemento y el último
- Los elementos de índice par
- Los elementos de índice impar

```
COMIENZO
vec(50)
PARA i = 1 A 50
    INGRESAR "ingrese un número", vec(i)
PROXIMO
IMPRIMIR "Cuarto elemento", vec(4)
IMPRIMIR "Segundo elemento", vec(2)
PARA i = 50 A 1 PASO -1
    IMPRIMIR vec(i)
PROXIMO
IMPRIMIR "Producto ", vec(1) * vec(50)
PARA i = 2 A 50 PASO 2
    IMPRIMIR vec(i)
PROXIMO
PARA i = 1 A 49 PASO 2
    IMPRIMIR vec(i)
PROXIMO
FIN
```

Se puede apreciar como es válido imprimir un elemento particular de un vector.

```
IMPRIMIR "Cuarto elemento", vec(4)
IMPRIMIR "Segundo elemento", vec(2)
```

También es correcto imprimir un vector en forma invertida o alternadamente para lo cual habrá que tener especial cuidado con la definición del PASO en el ciclo PARA.

PROBLEMAS Y EJERCICIOS DE APLICACIÓN

Ejercicio 2

Cargar un vector de n elementos con números enteros e imprimir:

- Cantidad de números positivos
- Cantidad de números negativos
- Cantidad de números menores de 25

COMIENZO

INGRESAR "Ingrese cantidad de elementos", n
 $vec(n) : pos = 0 \quad neg = 0 \quad menor25 = 0$

PARA $i = 1$ A n

INGRESAR "Ingrese un número", $vec(i)$

SI $vec(i) > 0$ ENTONCES $pos = pos + 1$

SI $vec(i) < 0$ ENTONCES $neg = neg + 1$

SI $vec(i) < 25$ ENTONCES $menor25 = menor25 + 1$

PROXIMO

IMPRIMIR "Cantidad de números positivos", pos

IMPRIMIR "Cantidad de números negativos", neg

IMPRIMIR "Cantidad de números menores a 25", $menor25$

FIN

Este ejercicio presente como variante que la cantidad de elementos del vector debe ser ingresada previo al dimensionamiento de mismo:

INGRESAR "Ingrese la cantidad de elementos", n
 $vec(n)$

Ejercicio 3

Dados 25 edades, cargarlas en un vector y calcular el promedio:

a) Cargar por un vector

b) Cantidad de edades mayores a 18 años

c) Todas las edades mayores al promedio

COMIENZO

$sum = 0$ mayor = 8 0

edad(25)

PARA $i = 1$ A 25

INGRESAR "Ingrese edad:", $edad(i)$

$sum = sum + edad(i)$

SI $edad(i) > 18$ ENTONCES $mayor = mayor + 1$

PROXIMO

$edprom = sum / 25$

IMPRIMIR "Edad promedio", $edprom$, "años"

IMPRIMIR "Cantidad de edades mayores a 18 años:", $mayor$ 8

IMPRIMIR "Edades mayores al promedio,"

PARA $i = 1$ A 25

SI $edad(i) > edprom$ ENTONCES IMPRIMIR $edad(i)$

PROXIMO

FIN

En este ejemplo se utilizó el vector como vector de almacenamiento de la suma.

$sum = sum + edad(i)$

También dada la cantidad de procesos, se podría haber realizado cada cálculo en forma independiente.

COMIENZO

$sum = 0$ mayor = 8 0

edad(25)

PARA $i = 1$ A 25

INGRESAR "Ingrese edad", $edad(i)$

PROXIMO

PARA $j = 1$ A 25

$sum = sum + edad(i)$

PROXIMO

$edprom = sum / 25$

PARA $i = 1$ A 25

SI $edad(i) > 18$ ENTONCES $mayor = mayor + 1$

PROXIMO

```

IMPRIMIR "Edad promedio", edprom, arcos
MPRIMIR "Cantidad de edades mayores a 18 años" mayor18
MPRIMIR "Edades mayores al promedio"
PARA i = 1 A 25
    SI edad(i) > edprom ENTONCES IMPRIMIR edad(i)
PROXIMO
FIN

```

La segunda forma de resolver el ejercicio será leyendo de estudio en el próximo capítulo.

Ejercicio 4

Una empresa desea proporcionar sueldos de sus empleados. Para cada empleado se conoce sueldo y nombre.

Se desea saber

- Nombre de los empleados con sueldo mayor al sueldo promedio.
- Cantidad de empleados que ganan más de 500\$

```

COMENZO
INGRESAR "Ingreso la cantidad de empleados", n
sueldo(n) con$(n)
total = 0 mas500 = 0
PARA i = 1 A n
    INGRESAR "Ingreso el nombre", nom$(i)
    INGRESAR "Ingreso el sueldo", sueldo(i)
PROXIMO
PARA i = 1 A n
    total = total + sueldo(i)
    SI sueldo(i) > 500 ENTONCES mas500 = mas500 + 1
PROXIMO
edprom = total / n
IMPRIMIR "Edades mayores al promedio"
PARA i = 1 A n
    SI sueldo(i) > edprom ENTONCES IMPRIMIR nom$(i)
PROXIMO
MPRIMIR "Cantidad de empleados que ganan más de 500$.", mas500
FIN

```

Este ejercicio nos presenta como variante, e imprimir el valor del vector `nom$(i)`, condicionada por el vector `sueldo(i)`.

Si `sueldo(i) > edprom` ENTONCES IMPRIMIR `nom$(i)`

5.6 MÁXIMOS Y MÍNIMOS DE UN VECTOR

Anteriormente nos referimos a la búsqueda de máximos y mínimos entre un conjunto de datos.

En los ejercicios desarrollados a continuación veremos, como se encuentra dicha búsqueda dentro de un vector.

Ejercicio 5

Dado un vector de 30 elementos, imprimir el valor máximo (suponemos que el vector fue cargado previamente).

```

COMENZO
vec(30)
vmax = vec(1)
PARA i = 2 A 30
    SI vec(i) > vmax ENTONCES vmax = vec(i)
PROXIMO
MPRIMIR "Valor máximo", vmax
FIN

```

La primera gran dificultad está en que no tenemos que colocar la variable `vmax` en cero, pues al disponer de todos los datos recordando al vector se define inicialmente a `vmax`, con el dato almacenado en el primer elemento `vmax = vec(1)`.

A continuación se recorre el vector desde el segundo elemento hasta el último, utilizando un ciclo PARA. Esta búsqueda permite encontrar el valor máximo buscado en el ejercicio.

Ejercicio 6

Dado un vector de 10 elementos, imprimir el valor mínimo (suponemos que el vector fue cargado previamente).

por considerar que a los fines de nuestra explicación, tener una buena relación entre la didáctica y la velocidad y eficiencia en su uso.

5.7.1 Método del Burbujeo

Se basa en llevar el más no del vector a la última posición. Para lograr esto se van tomando los elementos de a dos y se los va comparando e intercambiando hasta cumplir con el objetivo. Además para optimizar su funcionamiento utilizaremos un switch que nos permitirá conocer si el vector está ordenado o no.

Dado un vector de 10 elementos, vamos a ordenarlo en forma ascendente.

```
COMENZO
vec[0]
cola = 10
k = 1
HACER MIENTRAS k <> 0
  k = 0
  PARA i = 1 A cola - 1
    Si vec(i) > vec(i + 1) ENTONCES
      aux = vec(i)
      vec(i) = vec(i + 1)
      vec(i + 1) = aux
      k = 1
  FIN PARA
  PROXIMO
  cola = k
REPETIR
FIN
```

Primero vamos a asignar a la variable cola un valor igual a la cantidad de elementos del vector a ordenar; más adelante explicaremos su función. La variable k es un switch que cumple con las siguientes condiciones:

Si $k = 0$, implica que el vector está ordenado; si $k \neq 0$, el vector está desordenado.

Al comienzo de ejecución suponemos que el vector está desordenado, lo

que se indica en el código con el valor inicial de k = 1.

que indicamos colocarlo en 1 (k podría haber sido igual a cualquier valor distinto de cero).

Esto hace que nos aseguremos la entrada en el ciclo HACER MIENTRAS al preguntar por $k \neq 0$.

Una vez dentro de ciclo se coloca a k en 0. De esta manera estamos suponiendo que una vez que se ejecuten las instrucciones del ciclo HACER MIENTRAS, el vector quedará ordenado.

A continuación tenemos un ciclo PARA de valor inicial igual a uno y de valor final a cola - 1 ($\text{PARA } i = 1 \text{ A } \text{cola} - 1$). Lo cual coincide con la cantidad de elementos, menos uno, que tiene el vector.

Esto está relacionado con la primera instrucción dentro del ciclo PARA.

Si $\text{vec}(i) > \text{vec}(i + 1)$ ENTONCES, que compara el primer elemento con el segundo, el segundo con el tercero, el tercero con el cuarto, etc., hasta llegar a comparar los dos últimos elementos. Si el elemento está fuera de la cola, la última comparación la hará entre el elemento 0 ($i = \text{cola}$) y el inexistente elemento 1. Esto explica el motivo de recorrer hasta $\text{cola} - 1$.

Cada vez que la condición es verdadera, hay que hacer un intercambio de valores.

```
aux = vec(i)
vec(i) = vec(i + 1)
vec(i + 1) = aux
```

Como se puede apreciar se utiliza una variable auxiliar aux, para no perder ningún dato al copiar de $\text{vec}(i)$ a $\text{vec}(i + 1)$ y viceversa.

Luego aparece nuevamente una asignación a la variable k, esta vez del índice i.

```
k = i
```

El valor de i en el momento de la asignación corresponde a posición de los últimos elementos intercambiados, lo que tiene un doble significado:

✓ Cargar k con un valor distinto de cero para volver a ejecutar el ciclo HACER MIENTRAS, pues el hecho de haber ejecutado un intercambio condici-
ona a que se ejecute el ciclo al menos una vez más para verificar si el vector quedó ordenado.

✓ Transferir fuera del ciclo PARA, el valor de k hacia cola ($\text{cola} = k$), para no tener que volver a comparar nuevamente desde el primer elemento hasta

que se indica en el código con el valor inicial de k = 1.

Vectores

el último, sino hasta el último intercambiado. Se supone que el resto del vector no tuvo intercambio, porque está ordenado.

El vector estará ordenado cuando al ejecutar integralmente el ciclo PARA, nunca se cumpla la condición verificada por la instrucción SI ENTONCES. Si esto sucede no se asigna a k , por lo que al terminar esta última en cero, valor almacenado en k luego de ingresar en el ciclo HACER MIENTRAS. De esta forma no se cumplirá la condición $k < 0$. Y al final cuando el proceso de ordenamiento termina.

Supongamos que los elementos de la lista contengan los siguientes valores:

9 7 6 5 4 3 1 8 10 2

En una primera pasada por el ciclo PARA los elementos quedarán de la siguiente forma:

1 6 5 4 3 9 8 10 2 11

En la segunda pasada no hará falta comparar hasta el último elemento, porque en la anterior pasada ya hemos llevado el máximo a la última posición.

6 5 4 3 7 8 9 2 10 11

De esta forma procederemos con las siguientes pasadas:

5 4 3 6 7 8 2 9 10 11 sexta pasada.

4 3 5 6 7 2 8 9 10 11 cuarta pasada.

3 4 5 6 2 7 8 9 10 11 quinta pasada.

3 4 5 7 6 7 8 9 10 11 sexta pasada.

3 4 2 5 6 7 8 9 10 11 séptima pasada.

3 2 4 5 6 7 8 9 10 11 octava pasada.

2 3 4 5 6 7 8 9 10 11 novena pasada.

2 3 4 5 6 7 8 9 10 11 décima pasada.

772. Algoritmo de ordenamiento de datos.

Vectores

La décima pasada arroja como resultado la misma ubicación de elementos que la novena, lo cual indica que el vector está ordenado.

Ejercicio 8

Dado un vector de 24 elementos, ordenarlo en forma descendente

COMENZAR

vec(24)

cota = 24

k = 1

HACER MIENTRAS k <= 0

k = 0

PARA i = 1 A cota - 1

SI vec(i) < vec(i + 1) ENTONCES

aux = vec(i)

vec(i) = vec(i + 1)

vec(i + 1) = aux

k = i

PROXIMO

cota = k

REPETIR

FIN

Cuando ordenamos en forma descendente, debemos invertir la pregunta en la condición de la instrucción SI ENTONCES: $vec(i) < vec(i + 1)$. El resto del algoritmo es igual o explicado para el ordenamiento ascendente.

Ejercicio 9

Dados un vector sueldo y un vector nombre\$, cargarlos con los sueldos y nombres de 50 empleados y luego imprimir ambos, ordenados en forma ascendente por sueldo de empleado.

COMENZAR

sueldo(50) = nombre\$(50)

PARA i = 1 A 50

773. Algoritmo de ordenamiento de datos.

Vectores

INGRESAR " ingrese el nombre del empleado ", nombre\$(i)
INGRESAR " ingrese el sueldo del empleado ", sueldo\$(i)

PROXIMO

k = 1

HACER MIENTRAS k <= 0

k = 0

PARA i = 1 A tota - 1

Si sueldo(i) > sueldo(i + 1) ENTONCES

aux = sueldo(i)

sueldo(i) = sueldo(i + 1)

sueldo(i + 1) = aux

aux\$ = nombre\$(i)

nombre\$(i) = nombre\$(i + 1)

nombre\$(i + 1) = aux\$

k = i

FIN SI

PROXIMO

cont = k

REPETIR

FIN

En este ejemplo se trabaja con dos vectores paralelos sueldo que determina el ordenamiento y nombre\$ que trabaja asociado al vector sueldo. Ambos vectores se cargan paralelamente. Esto quiere decir que el nombre del empleado cargado en la posición i del vector nombre\$ le va a corresponder el sueldo cargado en la posición i del vector sueldo. El algoritmo de ordenamiento es el mismo que utilizamos en los ejemplos anteriores, con el agregado de tener que intercambiar el vector nombre\$ cada vez que corresponda intercambiar el vector sueldo.

Si sueldo(i) > sueldo(i + 1) ENTONCES

aux = sueldo(i)

sueldo(i) = sueldo(i + 1)

sueldo(i + 1) = aux

aux\$ = nombre\$(i)

nombre\$(i) = nombre\$(i + 1)

nombre\$(i + 1) = aux\$

k = i

FIN SI

Vectores

La variable auxiliar utilizada debe ser alfanumérica como nombre\$, aux\$.

RESUMEN

Estructura de datos es el conjunto de datos, organizados bajo un nombre común, que se pueden tratar como una sola unidad.

Las estructuras de datos, de acuerdo al lugar donde se almacenan, se clasifican en:

✗ Internas: se almacenan en la memoria de la computadora, se clasifican en vectores y matrices según el tipo.

✗ Externas: se almacenan en soportes externos como los discos; se denominan archivos.

✗ Un vector es un conjunto de datos del mismo tipo, números o alfanuméricos, organizados bajo un mismo nombre y almacenados en la memoria de la computadora.

✗ Cada dato se denomina elemento del vector y está referenciado por un variable índice, que indica la posición de dicho elemento dentro del vector.

✗ Previo al uso de un vector, hay que reservar una zona de memoria para su uso, así como definir el número de elementos necesarios para acceder a cada elemento de la estructura. Al proceso de reservar una zona de memoria para almacenar los datos de vector, se lo denomina "dimensionar" y se realiza de la siguiente forma:

Sintaxis:

Nombre del vector [cantidad de elementos]

✗ La posición es la dirección del lugar que ocupa cada dato dentro del vector y está indicado por el índice.

✗ El elemento es el dato que hay a almacenado en el vector, en esa posición.

Algoritmos y estructuras de datos

VECTORES

- Los vectores, al residir en la memoria de la computadora, son de acceso directo, por lo que podemos situarnos directamente en la posición deseada
- Los elementos de un vector se utilizan en forma análoga a la de cualquier variable
- Los vectores que contienen datos alfanuméricos, deben escribirse con el símbolo \$ al final: vector(i)

Máximos y mínimos: al disponer de todos los datos, invocando al vector, se define inicialmente a vmax, con el dato almacenado en el primer elemento vmax = vec(1)
A continuación se recorre el vector desde el segundo elemento hasta el último, utilizando un ciclo PARA. Esta búsqueda permite encontrar el valor máximo buscado
El algoritmo para calcular el máximo de un vector llamado vec(n) es

```
vec(n)
max = vec(1)
PARA i = 2 A n
    S vec(i) > max ENTONCES max = vec(i)
PROXIMO
```

Donde n es la cantidad de elementos del vector
El algoritmo para un mínimo sería

```
vec(n)
min = vec(1)
PARA i = 2 A n
    S vec(i) < min ENTONCES min = vec(i)
PROXIMO
```

En los procedimientos utilizados aquí, todos utilizados antes, empleamos como restricción, que no podían existir máximos y mínimos múltiples, sólo un valor entre un conjunto podría estar asociado a resultado de la búsqueda

Una gran ventaja en la utilización de vectores es la posibilidad de poder encontrar más de una variable asociada a un máximo o a un mínimo, que responde a la búsqueda solicitada. Esto es así cuando se trabaja con dos o más vectores asociados

Si bien el valor máximo o mínimo sigue siendo uno sólo, al ser múltiple la

117 - ALGORITMOS Y ESTRUCTURAS DE DATOS

VECTORES

posibilidad de encontrar varios elementos en vectores asociados al de búsqueda, se determinan a estos valores asociados, hallados, como "máximos y mínimos múltiples"

La búsqueda se hace en dos etapas: primero se determina el máximo en el vector de búsqueda se buscan todos los elementos asociados a este valor en los otros vectores

Ordenamiento: El ordenamiento puede ser:

/ **Ascendente:** los elementos están situados de menor a mayor
/ **Descendente:** los elementos están situados de mayor a menor.

Los valores repetidos, en caso de existir, quedan en posiciones contiguas. Utilizamos el método del burbujeo que se basa en llevar el máximo del vector a la última posición. Para lograr esto se varía tomando los elementos de a dos y se los va comparando e intercambiando hasta cumplir con el objetivo. Además para optimizar su funcionamiento se utiliza un switch que nos permite recorrer el vector está ordenado o no.
El algoritmo es el siguiente

```
cota = cantidad de elementos a ordenar
i = 1
HACER MIENTRAS k <> 0
    k = 0
    PARA i = 1 A cota - 1
        S vec(i) > vec(i + 1) ENTONCES
            aux = vec(i)
            vec(i) = vec(i + 1)
            vec(i + 1) = aux
            k = i
    FIN SI
    PROXIMO
    cota = k
FINTE
```

La variable k es un switch que cumple con las siguientes condiciones

Si k = 0, implica que el vector está ordenado; si k <> 0 el vector está desordenado.

Al comienzo del ejercicio suponeremos que el vector está desordenado, lo

ALGORITMOS Y ESTRUCTURAS DE DATOS - 117

Vectores

que indicamos cooccurda k en l (k pod'a habers do igual a qu'quier valor distinto de cero)

Esto hace que nos aseguremos la entrada en el ciclo HACERMIENTRAS al preguntar por k <> 0

Una vez dentro de ciclo se coloca a k en 0. De esta manera estamos suponiendo que una vez más se ejecuten las instrucciones de ciclo HACERMIENTRAS, el valor quedará ordenado

A continuación tenemos un ciclo PARA de valor inicial igual a uno y de valor final a cota (PARA i = 1 A cota-l), lo cual coincide con la cantidad de elementos, menos uno, que tiene el vector.

Esto está relacionado con la primera construcción dentro del ciclo PARA.

Si $vec(i) > vec(i + 1)$ ENTONCES, que compara el primer elemento con el segundo, el segundo con el tercero, el tercero con el cuarto, etc, hasta llegar a comparar los dos últimos elementos. Si el ciclo fuera de l a cota, la última comparación la haría entre el último elemento (l - cota) y un elemento "existente", $i + 1 - cota + 1$. Esto explica el motivo de recorrer hasta cota - l

Cada vez que la condición es verdadera, hay que hacer un intercambio de valores.

```
aux = vec(i)
vec(i) = vec(i + 1)
vec(i + 1) = aux
```

Como se puede apreciar se utiliza una variable auxiliar aux, para no perder ningún dato al copiar de $vec(i)$ a $vec(i + 1)$ y viceversa

Luego aparece nuevamente una asignación a variable k, esta vez del índice i

$i = i + 1$

El valor de i es el momento de la asignación corresponde a la posición de los últimos elementos intercambiados, o que tiene un doble significado.

✗ Cargar k con un valor distinto de cero para volver a ejecutar el ciclo HACERMIENTRAS, pues es hecho de haber realizado un intercambio condicional a que se ejecute el ciclo al menos una vez más para verificar si el vector quedó ordenado

✗ Transferir fuera del ciclo PARA, el valor de k hacia cota (cota = k), para no tener que volver a comparar nuevamente desde el primer elemento hasta

Vectores

el último, sino hasta el último intercambiado. Se supone que el resto del vector no tuvo intercambios porque está ordenado

El vector estará ordenado cuando al ejecutar integralmente el ciclo PARA, nunca se cumpla la condición verificada por la instrucción Si ENTONCES. Si eso sucede no se asignará a k, permaneciendo esta última en cero, valor cargado en k luego de ingresar en el ciclo HACERMIENTRAS. De esta forma no se cumplirá la condición k <> 0 y finalizará el proceso de ordenamiento

Ejercicios Propuestos

1. Cargar un vector de 45 elementos e imprimir el valor de cada uno de estos
2. Dados 20 números, cargarlos en un vector y tallar e imprimir:
 - a) la suma de los elementos.
 - b) la cantidad de elementos del vector iguales a 1
3. Cargar un vector de n elementos y luego imprimir:
 - a) los elementos pares.
 - b) la suma de los elementos
 - c) El promedio de los elementos.
 - d) El porcentaje de elementos positivos
4. Dados los sueldos y edades de 1 empleado, de una empresa, se pide cargarlos en vectores y luego imprimir:
 - a) Sueldo promedio
 - b) Sueldo promedio de los empleados que tengan entre 18 y 20 años
 - c) Edad promedio.
 - d) Cantidad de empleados con sueldo mayor al sueldo promedio
 - e) Cantidad de empleados con edad menor a la edad promedio
5. Dadas n notas y edades de alumnos, se pide cargarlos en vectores y luego imprimir:
 - a) Cantidad de alumnos aprobados
 - b) Cantidad de alumnos calificados
 - c) Edad promedio.
 - d) Nota promedio de los alumnos mayores a 15 años
5. Se tiene un vector cargado con 100 números. Se pide tallar e imprimir,

6. Se tiene un vector cargado con 100 números. Se pide tallar e imprimir, la cantidad de números positivos, negativos y ceros.
7. Se deben cargar en un vector los tiempos de clasificación de 60 autos los autos se identifican con números correlativos del 1 al 60. Se pide imprimir:
 - a) Número de auto que clasificó primero.
 - b) Porcentaje de clasificación
8. Dado el vector tiempos del ejercicio 7, ordenarlo en forma ascendente. Se deberá imprimir de la siguiente forma.

Número de auto	Tiempo
----------------	--------

Subprogramas

6.1 INTRODUCCIÓN

Los programas realizados hasta ahora están todos desarrollados dentro de un único programa denominado **programa principal**.

En el capítulo 1, cuando hablamos de las características de los algoritmos, hicimos referencia al concepto de modularidad, que consiste en estructurar el programa principal en módulos más pequeños llamados **subprogramas**.

Un subprograma es un conjunto de instrucciones de un programa que llevan a cabo una determinada tarea y que puede ejecutarse desde distintos puntos del programa principal.

Al finalizar la ejecución del subprograma se regresa a punto de partida del programa principal continuando la secuencia de ejecución de esa estructura de un subprograma es básicamente la de cualquier programa, con las diferencias lógicas en el comienzo y el fin.

Un subprograma puede a su vez estar compuesto por varios subprogramas más descriptos fuera del programa principal o estructura es la siguiente

```
COMIENZO
NOMBRESUBPROGRAMA 1
NOMBRESUBPROGRAMA 2
NOMBRESUBPROGRAMA 3
NOMBRESUBPROGRAMA N
FIN

NOMBRESUBPROGRAMA 1.
Instrucciones
```

RETORNAR
NOMBRESUBPROGRAMA 2.

Instrucciones

REICRNAR
NOMBRESUBPROGRAMA 3

Instrucciones

RETORNAR

NOMBRESUBPROGRAMA N

Instrucciones

RETORNAR

Los nombres de subprogramas los escribiremos en mayúscula, para una mayor legibilidad del programa. Pueden utilizarse letras, números y guiones pero no deben darse espacios en blanco y siempre el primer carácter debe ser una letra.

En la llamada al subprograma dentro del programa principal, se coloca sólo el nombre. En el subprograma previamente dicho, acompañaremos el nombre con dos puntos (:) al final y luego de todas las instrucciones indicamos el regreso al programa principal con la palabra RETORNAR.

La cantidad de subprogramas que habrá dentro de cada programa será determinada por el programador en función de la complejidad del ejercicio.

El objetivo de cumplir por los subprogramas es de conseguir una mayor estructura del programa, facilitando su construcción y simplificarlo al máximo las futuras modificaciones del programa.

Una gran ventaja de utilizar subprogramas, es de evitar repeticiones de instrucciones dentro de un programa, estas instrucciones se escribirán en un subprograma al cual es llamado y ejecutado las veces que haga falta.

De esta manera las instrucciones están escritas una sola vez, ocupando menos memoria en el almacenamiento del programa.

Subprogramas e Instrucciones de datos

6.2 EJERCICIOS DE APLICACIÓN

Vamos a desarrollar un conjunto de ejercicios que nos permitan y simplificar por un lado los conceptos vistos hasta ahora, y además comenzar a trabajar dividiendo el programa en un conjunto de subprogramas.

Ejercicio 1

Una editorial de libros posee 12 libros diferentes numerados del 1 al 12. Cuenta con una planilla en la que figuran los siguientes datos:

Número de libro.
Título.
Autor.
Costo.
Precio de venta.
Cantidad de ejemplares vendidos.

Se pide calcular e imprimir:

- Título del libro con mayor cantidad de ejemplares vendidos.
- Autor y título del libro con menor costo.
- Facturación total de la editorial.
- Ganancia de la editorial.

COMIENZO
INICIO
CARGA.
MAXIMO
MINIMO
FACTURACION
GANANCIAS
IMPRESION
FIN

INICIO.
numero(12). titulo\$(12) autor\$(12) costo(12) precio(12) cantidad(12)
RETORNAR

Subprogramas e Instrucciones de datos

```

CARGA
PARA i = 1 A 2
  INGRESAR "Número de libro", numero(i)
  INGRESAR "Título", titulo$(i)
  INGRESAR "Autor", autor$(i)
  INGRESAR "Costo", costo(i)
  INGRESAR "Precio de venta", precio(i)
  INGRESAR "Cantidad de ejemplares vendidos", cantidad(i)
PROXIMO
RETORNAR

MAXIMO
max = cantidad(1)
PARA i = 2 A 12
  Si cantidad(i) > max ENTONCES max = cantidad(i)
PROXIMO
RETORNAR

MINIMO
min = costo(1)
PARA i = 2 A 12
  Si costo(i) < min ENTONCES min = costo(i)
PROXIMO
RETORNAR

FACTURACION
fact = 0
PARA i = 1 A 12
  fact = fact + precio(i)
PROXIMO
RETORNAR

GANANCIAS
PARA i = 1 A 12
  sumcosto = sumcosto + costo(i)
PROXIMO
ganancia = fact - sumcosto
RETORNAR

```

```

IMPRESION:
IMPRIMIR "Libro de mayor cantidad de ejemplares vendidos"
PARA i = 1 A 2
  Si cantidad(i) = max ENTONCES IMPRIMIR titulo$(i)
PROXIMO
IMPRIMIR "Libro de menor costo"
PARA i = 1 A 12
  Si costo(i) = min ENTONCES IMPRIMIR autor$(i), titulo$(i)
PROXIMO
IMPRIMIR "Facturación total.", fact, "$"
IMPRIMIR "Ganancias", ganancia, "$"
RETORNAR

```

Se diseñó el programa en siete subprogramas: INICIO, CARGA, MAXIMO, MINIMO, FACTURACION, GANANCIAS e IMPRESION, recordando que los nombres de subprogramas no llevan acento al igual que las variables. En INICIO se diseñaron los 6 vectores para cargarlos luego en CARGA.

Los cálculos fueron realizados en los cuatro subprogramas siguientes: MAXIMO, MINIMO, FACTURACION y GANANCIAS. Los resultados se muestran todos en el subprograma IMPRESION.

Ejercicio 2

Una empresa maneja sus ventas por medio de corredores, cada uno de ellos cobra un porcentaje sobre lo vendido.

Además el corredor que vende más de 5000\$ recibe un premio de 2000\$ si el descuento.

Los corredores son 7 y los porcentajes de comisión son:

1 y 2:	10%
3 y 5:	5%
4:	7%
6 y 7:	3%

Para cada corredor se conoce además el nombre y el porcentaje. Se desea calcular e imprimir:

a) Ventas totales de la empresa

Subprogramas

```

S comision(i) < min ENTONCES m = comision(i)
PROXIMO
RETURNAR

IMPRESION
IMPR MIR "Veras totales de la empresa:", toverta, "$"
IMPR MIR "Veras por corredor"
PARA i = 1 A 7
    IMPR MIR "Corredor:", i, "Veras:", venta(i), "$"
PROXIMO
IMPR MIR "Pedidos por corredor:"
PARA i = 1 A 7
    IMPR MIR "Corredor:", i, "Pedidos:", pedido(i)
PROXIMO
IMPR MIR "Comisór por corredor:"
PARA i = 1 A 7
    IMPR MIR "Corredor:", i, "Comisór:", comision(i), "$"
PROXIMO
IMPR MIR "Porcentaje de ventas de cada corredor"
PARA i = 1 A 7
    IMPR MIR "Corredor:", i, "Porcentaje:", porcent(i), "%"
PROXIMO
IMPR MIR "Corredores de mayor venta"
PARA i = 1 A 7
    SI venta(i) = max FNCION(LS MPR MIR "Corredor:",
PROXIMO
MIR MIR "Nombres de los corredores con mayor comisór"
PARA i = 1 A 7
    SI comision(i) = max FNCION(LS IMPR MIR "Nombre:", i)
PROXIMO
RETURNAR
    
```

Se diseñó el programa en 7 subprogramas, nuevamente INICIO y CARGA al comienzo, luego los distintos procesos de cálculo: COMISION, PORCENTAJE, MAXIMO y MINIMO, finalizando con el subprograma de impresión.

Se dimensionaron 6 vectores de los cuales venta() y pedido() se colocaron en cero por ser un acumulador y un contador respectivamente. Es interesante ver como se inicializó comision(), con los distintos porcentajes de comisión de acuerdo al número de corredor.

Algoritmos y procedimientos de desarrollo

Subprogramas

En la carga, no sólo se ingresaron los datos, además se cargaron los vectores venta() y pedido() a comision es un vector que se carga con el producto entre dos vectores venta() y comision().

El porcentaje como siempre está dado por el producto entre el valor individual y el total por 100.

los subprogramas de máximo, mínimo e impresión son similares a los vistos en ejercicios anteriores.

Ejercicio 3

Una empresa de informática tiene 100 empleados. Cada uno de ellos pertenece a una categoría que se le asigna cuando ingresa a trabajar en la empresa.

Las categorías son cinco:

- 1 Analista Senior
- 2 Analista Junior
- 3 Programador Senior
- 4 Programador Junior
- 5 Operador

Los sueldos son

- 1 2500\$
- 2 2000\$
- 3 1500\$
- 4 1200\$
- 5 800\$

El empleado puede trabajar en dichas categorías en alguno de los tres departamentos que posee la empresa. Los sueldos varían fijos para cada categoría.

A fin de mes se liquidan los sueldos en una planilla donde figura:

- Nombre del empleado.
- Categoría.
- Departamento (1, 2 y 3)

Se pide calcular e imprimir

Algoritmos y procedimientos de desarrollo

Subproceso

- 1) Cantidad de empleados por categoría
- 2) Sueldos totales de cada categoría
- 3) Cantidad de empleados por departamento
- 4) Nombre de la categoría que tenga más empleados
- 5) Número de departamento que tenga la mínima cantidad de empleados.
- 6) Imprimir en forma ordenada ascendente por sueldos de cada categoría.
- 7) Sueldo total.
- 8) Cantidad de empleados.

COMIENZO
 INICIO
 CARGA
 MAXIMO
 MINIMO
 ORDEN
 IMPRESION
 FIN

```

INICIO
empcat(5) sdocat(5) empdep(3) . nombre$(5) . sueldo(5) . numero(5)
nombre$(1) = "Analista Senior"
nombre$(2) = "Analista Junior"
nombre$(3) = "Programador Senior"
nombre$(4) = "Programador Junior"
nombre$(5) = "Operador"
sueldo(1) = 2500
sueldo(2) = 2000
sueldo(3) = 1500
sueldo(4) = 1200
sueldo(5) = 800
PARA i = 1 A 5
  numero(i) = i
  empcat(i) = 0
  sdocat(i) = 0
  
```

Subproceso

```

PROXIMO
PARA i = 1 A 3
  empdep(i) = 0
PROXIMO
RETORNAR
  
```

```

CARGA
PARA i = 1 A 100
  INGRESAR "Categoría", cat
  INGRESAR "Número de departamento", dep
  empcat(cat) = empcat(cat) + 1
  empdep(dep) = empdep(dep) + 1
  sdocat(cat) = sdocat(cat) + sueldo(cat)
PROXIMO
RETORNAR
  
```

```

MAXIMO:
max = empcat(1)
PARA i = 2 A 5
  Si empcat(i) > max EN ONCES max = empcat(i)
PROXIMO
RETORNAR
  
```

```

MINIMO
min = empdep(1)
PARA i = 2 A 3
  Si empdep(i) < min EN ONCES min = empdep(i)
PROXIMO
RETORNAR
  
```

```

ORDEN
cola = 5
i = 1
HACER MIENTRAS k <> 0
  k = 0
  PARA j = 1 A cola - 1
    Si sdocat(j) > sdocat(j + 1) EN ONCES
      aux = sdocat(j)
      sdocat(j) = sdocat(j + 1)
      
```

Subprogramas

```

sdcacat (i + 1) = aux
aux = numero (i)
numero (i) = numero (i + 1)
numero (i + 1) = aux
aux$ = nombre$ (i)
nombre$ (i) = nombre$ (i + 1)
nombre$ (i + 1) = aux$
aux = empcat (i)
empcat (i) = empcat (i + 1)
empcat (i + 1) = aux

```

k i

FIN SI

PROXIMO

con k

REPTIR

RETORNAR

IMPRESION

IMPRMIR "Cantidad de empleados por departamento"

PARA i = 1 A 3

IMPRMIR "Departamento ", "Empleados.", empdep(i)

PROXIMO

IMPRMIR "Categoría con más empleados"

PARA i = 1 A 3

Si empcat(i) > max ENTONCES IMPRIMIR nombre\$(i)

PROXIMO

IMPRMIR "Departamentos con mayor cantidad de empleados"

PARA i = 1 A 3

Si empdep(i) > max ENTONCES IMPRIMIR "Departamento",

PROXIMO

IMPRMIR "Sueldos por categoría:"

PARA i = 1 A 3

IMPRMIR numero(i)

IMPRMIR nombre\$(i)

IMPRMIR "sueldo", sdcacat(i), "\$"

IMPRMIR "Cantidad de empleados", empcat(i)

PROXIMO

RETORNAR

134. Subprogramas y procedimientos de datos

Subprogramas

Se dividió el programa en 6 subprogramas: INICIO, CARGA, MAXIMO, MINIMO, ORDEN e IMPRESION

Todos los subprogramas, dos de los cuales son contadores: empcat(i) y empdep(i); y un acumulador: sdcacat(i).

En este caso tenemos como novedad la utilización de subprograma ORDEN que lleva a cabo el ordenamiento. El vector que ordena es el de sueldos por categoría: sdcacat(i) y trabajamos con todos a él y por lo tanto deben cambiar sus elementos, los vectores: numero(i), nombre\$(i), empcat(i)

Ejercicio 4

Una aerolínea comercial vende pasajes a cuatro destinos del exterior del país:

1. París
2. Roma.
3. Madrid
4. Londres

Los aviones tienen tres clases:

- | | |
|-------------|--------|
| 1. Primera | 1800\$ |
| 2. Negocios | 1600\$ |
| 3. Turista. | 900\$ |

Si el destino es Roma y viaja en primera, se le descuenta el 10% del precio total.

Se pide calcular el importe

- a) Cantidad de pasajes vendidos a cada destino
- b) Recaudación por cada destino
- c) Porcentaje de ventas en cantidad de pasajes de cada destino
- d) Cantidad de ventas en cantidad de pasajes de cada destino
- e) Nombre de la clase con mayor recaudación
- f) Ingresos pagados por la empresa (15% de la recaudación total)
- g) Número de destino con menor cantidad de pasajes vendidos

Los ítems a, b y c se deberán imprimir ordenados de manera ascendente por recaudación de cada destino de la siguiente forma

135. Subprogramas y procedimientos de datos

Número de destino
Recaudación por destino.
Porcentaje de ventas de cada destino.
Cantidad de pasajes vendidos por destino

COMIENZO

NICIO

PARA

PORCENTAJE

MAXIMO

MINIMO

MPUESTOS

ORDEN

MPRESION

FIN

NICIO:

numero(4) . recdest(4) pasdest(4) porcer(4)

pasclase(3) . recclase(3) . nombrc(3)

PARA i = 1 A 4

numero(i) - 1

recdest(i) - 0

pasdest(i) - 0

PROXIMO

PARA i = 1 A 3

recclase(i) - 0

pasclase(i) - 0

PROXIMO

nombrc(1) - 'Primero'

nombrc(2) - 'Negocios'

nombrc(3) - 'Turista'

RETORNAR

CARGA.

NGRESAR 'Número de pasaje', np

HACER MENIRAS np <> 0

NGRESAR 'Número de destino.', nd

NGRESAR 'Número de clase.', nc

SELECCIONAR CASO nc

CASO - 1

Si nd = 2 ENTONCES vp = 1800 *

O SI NO vp = 800

CASO 2

vp = 1600

CASO - 3

vp 900

FIN SELECCIONAR

pasdest(nd) pasdest(nd) + 1

recdest(nd) recdest(nd) + vp

pasclase(nc) pasclase(nc) + 1

recclase(nc) recclase(nc) + vp

REPETIR

RETORNAR

PORCENTAJE:

totpas - 0

PARA i = 1 A 4

totpas = totpas + pasdest(i)

PROXIMO

PARA i = 1 A 4

porcent(i) = pasdest(i) / totpas * 100

PROXIMO

RETORNAR

MAXIMO:

max = recclase(1)

PARA i = 2 A 3

Si recclase(i) > max ENTONCES max = recclase(i)

PROXIMO

RETORNAR

MINIMO

min = pasdest(1)

PARA i = 2 A 4

Si pasdest(i) < min ENTONCES min = pasdest(i)

PROXIMO

RETORNAR

Subprogramas

```

MPU STOS
*ofec = 0
PARA i = 1 A 3
    *ofrec = *ofrec + *recfase(i)
PROXIMO
impuesto *ofrec * 0.5
RETORNAR

ORDEN
cola = 4
k = 1
HACER MIENTRAS k <> 0
    k = 0
    PARA i = 1 A cola
        S recdes(i) > *recdes(i + 1) ENTONCES
            aux = recdes(i)
            *recdes(i) = recdes(i + 1)
            recdes(i + 1) = aux
            aux = numero(i)
            numero(i) = numero(i + 1)
            numero(i + 1) = aux
            aux = percent(i)
            percent(i) = percent(i + 1)
            percent(i + 1) = aux
            aux = pasdes(i)
            pasdes(i) = pasdes(i + 1)
            pasdes(i + 1) = aux
            k = 1
    PROXIMO
FIN SI

col = k
RETORNAR

IMPR:SION:
PARA i = 1 A 4
    IMPRIMIR "Número de destino", numero(i)
    IMPRIMIR "Recaudación:", recdes(i), "$"
    IMPRIMIR "Porcentaje de ventas", percent(i), "%"

```

Subprogramas

```

IMPRIMIR "Cantidad de pasajes vendidos:", pasdes(i)
PROXIMO
IMPRIMIR "Ventas por clase"
PARA i = 1 A 3
    IMPRIMIR "Clase:", nombre$(i)
    IMPRIMIR "Pasajes vendidos", pasase(i)
PROXIMO
IMPRIMIR "Clase de mayor recaudación:"
PARA i = 1 A 3
    Si recfase(i) = max ENTONCES IMPRIMIR nombre$(i)
PROXIMO
IMPRIMIR "Número de destino de menor recaudación:"
PARA i = 1 A 4
    Si pasdes(i) = min ENTONCES IMPRIMIR numero(i)
    IMPRIMIR "Puestos a pagar", impuesto, "$"
RETORNAR

```

El programa está dividido en 8 subprogramas: INICIO, CARGA, FOR, CTN'ALE, MAXIMO, MINIMO, IMPUESTOS, ORDEN e IMPRESION. En todos estos algoritmos ya fueron introducidos con diferentes variables en ejercicios anteriores.

Se utilizaron 7 vectores, de los cuales 2 son acumulados: recdes(i) y recfase(i), y dos contadores: pasdes(i) y pasfase(i).

Es interesante observar como en la carga se utilizó un instrucción de SE

LECCIONAR CASO, para determinar el valor de pasa e (vp) a acumular en

Los subprogramas de cálculo de recaudación por destino y por clase de los algoritmos clásicos estudiados.

El subprograma de impuesto es un cálculo porcentual directo, del total recaudado.

En el ordenar esto tenemos cuatro vectores numéricos intercambiando elementos, en función de valor de recaudación por destino recdes(i).

RESUMEN

Un subprograma es un conjunto de instrucciones de un programa que llevan a cabo una determinada tarea y que puede ejecutarse desde distintos puntos de programa principal.

Al finalizar la ejecución del subprograma se regresa al punto de partida del programa principal, continuando la secuencia de ejecución de este.

La estructura de un subprograma es básicamente la de cualquier programa, con las diferencias lógicas en el comienzo y el fin.

Un subprograma puede estar compuesto por varios subprogramas. Están descriptos fuera del programa principal.

La estructura a seguir es:

```
COMENZAR
NOMBRESUBPROGRAMA 1
NOMBRESUBPROGRAMA 2
NOMBRESUBPROGRAMA 3
.....
NOMBRESUBPROGRAMA N
FIN
```

NOMBRESUBPROGRAMA 1:

Instrucciones

RETORNAR

NOMBRESUBPROGRAMA 2:

Instrucciones

RETORNAR

NOMBRESUBPROGRAMA 3:

140 SUBPROGRAMAS Y ESTRUCTURAS DE DATOS

Instrucciones

RETORNAR

NOMBRESUBPROGRAMA N

Instrucciones

RETORNAR

El objetivo a cumplir por los subprogramas es el de conseguir una mayor estructuración de programas, facilitando su construcción y simplificación al máximo de futuras modificaciones del programa.

Una gran ventaja de utilizar subprogramas, es la de evitar repeticiones de instrucciones dentro de un programa; estas instrucciones se escriben en un subprograma el cual es llamado y ejecutado las veces que haga falta.

De esta manera las instrucciones están escritas una sola vez, ocupando menos memoria en el almacenamiento del programa.

141 SUBPROGRAMAS Y ESTRUCTURAS DE DATOS



Ejercicios Propuestos

1. La administración del estado posee 1200 empleados. Cada uno de ellos posee una categoría. Las categorías son tres. Los diferentes empleados se distribuyen en las siguientes dependencias:

1. Trabajo
2. Educación
3. Economía
4. Interior

Los sueldos por categoría son:

1. 600
2. 500
3. 400

El empleado puede trabajar en dichas categorías en alguna de las cuatro dependencias.

Los sueldos son fijos para cada categoría.

A fin de mes se liquidan los sueldos en una planilla donde figura:

Nombre del empleado
Categoría
Dependencia

Se pide calcular e imprimir:

- a) Cantidad de empleado por dependencia
- b) Sueldos totales de cada categoría
- c) Cantidad de empleados por dependencia
- d) Nombre de la categoría que tenga menos empleados
- e) Número de dependencia que haya pagado menos sueldos en pesos
- f) Imprimir en forma ordenada ascendente por sueldos de cada dependencia

LIBERTAD Y JUSTICIA

S u b p r o g r a m a s

dependencia, un listado con los siguientes datos:

Número de dependencia
Nombre de la dependencia
Sueldo total de la dependencia
Cantidad de empleados de la dependencia.

2. Una empresa transportadora de caudales posee 4 camiones blindados para realizar sus viajes. Cada camión puede transportar 3 tipos diferentes de valores:

1. Dólares
2. Pesos
3. Monedas de Oro

Cada camión transporta solamente uno de los tres tipos de valores en un viaje.

Dependiendo del tipo de valor que transporta el camión, es el precio que cobra:

1. 300 por viaje
2. 200 por viaje
3. 500 por viaje

Estos valores son independientes del camión que realiza el viaje.

A fin de mes la empresa efectúa una planilla con los siguientes datos:

Nombre del cliente
Tipo de valor
Camión que realizó el viaje

Se pide calcular e imprimir:

- a) Cantidad de viajes de cada camión
- b) Recaudación de cada tipo de valor
- c) Recaudación por cada camión
- d) Nombre del valor que se haya transportado más veces
- e) Número de camión que haya recaudado menos dinero
- f) Imprimir ordenado de manera ascendente por recaudación de tipo de

LIBERTAD Y JUSTICIA

valor, los siguientes datos

- Número de valor.
- Nombre de valor
- Recaudación por valor
- Cantidad de veces que se transportó cada valor

3. Una empresa transporta 4 centros para entregar y vender sus productos

1. Capital
2. Mendoza
3. Córdoba
4. Tucumán

En cada centro se entregan 5 productos diferentes los precios de los productos son

1. 0.80
2. 1.00
3. 0.35
4. 0.50
5. 0.70

A fin de mes la empresa confecciona una planilla con los datos de las ventas

- Número de cliente
- Número de centro
- Número de producto
- Cantidad de unidades

En 2000 ventas registradas en el mes Se pide calcular e imprimir

- a) Cantidad de ventas de cada centro.
- b) Recaudación de cada producto
- c) Recaudación de cada centro.
- d) Nombre de centro que recauda menos
- e) Número del producto que se vendió más en cantidad de unidades.

PROGRAMAS Y SISTEMAS DE DATOS

f) Imprimir ordenado en forma ascendente por recaudación de cada centro, los siguientes datos

- Número del centro
- Nombre del centro
- Recaudación del centro

7. Una empresa de combustible tiene 20 tipos de combustible que se vende

1. Nafta común
2. Nafta especial
3. Gas Oil

- 0.85 \$ por litro
- 1.00 \$ por litro.
- 0.50 \$ por litro

Cada vez que un automovilista carga alguno de los tres combustibles, se emite una boleta con los siguientes datos

- Número de boleta
- Número de estación (del 1 al 20)
- Tipo de combustible (1, 2 ó 3)
- Cantidad de litros

A fin de mes se procesan las ventas de todas las estaciones, teniendo en cuenta cada una de las boletas emitidas. El ingreso de datos finaliza con un número de boleta igual a cero

Los nombres de las estaciones se deberán cargar en un vector en memoria

Se pide calcular e imprimir

- a) Cantidad de litros vendidos en cada estación
- b) Recaudación de cada estación
- c) Cantidad de boletas emitidas por cada estación
- d) Total de litros vendidos para cada tipo de combustible
- e) Número de estación que recauda más
- f) Número de boleta con mayor importe.
- g) Impuestos pagados (10% de la recaudación total)

Los items a, b y c se deberán imprimir ordenados en forma ascendente, por recaudación por estación, de la siguiente manera

PROGRAMAS Y SISTEMAS DE DATOS

- 5 Una empresa envasadora de productos corrientes tiene 10 sucursales. En cada una de ellas se vende

- Cada vez que un mayorista compra alguno de los cuatro productos mencionados, se emite una boleta con los siguientes datos

- A fin de mes se procesaran las ventas de todas las sucursales, teniendo en cuenta cada una de las boletas emitidas. El ingreso de datos finalizará con un número de boleta igual a cero.
- Los nombres de las sucursales se deberán cargar en un vector en memoria, el cual guiará los nombres de las producciones.
- Se podrá calcular el \max y \min .

- Los ítems a, b, y c se deberán imprimir ordenados en forma descendente por recaudación de cada sucursal de la siguiente manera:

- 6 Una empresa de microtours vende pasajes a tres destinos del interior del país

- Siendo un proyecto piloto en materia de transporte automotor, los microbuses caseros

- El costo del bazo es el mismo para los tres destinos. Al pasajero se le entrega un ticket donde se consignar los siguientes datos.

- Si una persona compra más de 10 pasajes juntos, se le descuenta el 10% del precio total.

- A fin de mes se procesan las ventas de todos los destinos, teniendo en cuenta cada uno de los tickets emitidos. El ingreso de datos finaliza con un número de ticket igual a cero.

- Se pide calcular e imprimir:

- Cantidad de pasajes vendidos a cada des^{to} no.
- Recaudación por cada des^{to} no.
- Porcentaje de ventas, en cantidad de pasajes, de cada destino.

Subprogramas

- d) Cantidad de pasajes vendidos por caso
- e) Número de ticket con mayor importe en la venta
- f) Destino de menor recaudación.

Los items a, b, y c se deberán imprimir ordenados de manera ascendente por recaudación de cada destino de la siguiente forma:

- Número de destino
- Recaudación por destino
- Porcentaje de ventas de cada destino
- Cantidad de pasajes vendidos por destino

7. Una entidad de beneficencia, que autorizados 30 puestos de venta de panes en las diferentes asociaciones de ferrocarriles. En cada una de ellas se vende

Panetos	1.0\$
La burguesa	1.5\$
Basoosa	0.8\$

Cada vez que una persona compra alguno de los tres artículos, se emite un ticket con los siguientes datos:

- Número de ticket
- Número de puesto
- Total gastado.

A fin de mas se procesan las ventas de todos los puestos, teniendo en cuenta cada uno de los tickets emitidos. El ingreso de datos se realiza con un número de ticket igual a cero.

Los nombres de las estructuras se deberán seguir un convenio de nomenclatura.

Se pide calcular e imprimir:

- e) Recaudación de cada puesto
- b) Cantidad de tickets emitidos por cada puesto
- d) Cantidad de tickets con importe mayor a 10\$
- c) Número de puesto con mayor recaudación.
- f) Número de ticket con mayor importe.

Algoritmos y estructuras de datos

Subprogramas

- f) Números de puesto con recaudaciones menores a 1500\$

Los items c y b se deberán imprimir ordenados en forma descendente por recaudación de cada puesto de la siguiente manera:

- Número de puesto
- Recaudación por puesto
- Cantidad de tickets vendidos

Algoritmos y estructuras de datos

Matrices

7.1 INTRODUCCIÓN

Una matriz es un conjunto de datos homogéneos organizados bajo un mismo nombre, cada uno de los cuales debe referencarse por dos índices, i y j , también llamados subíndices, que denotarán las filas y las columnas de la matriz $i \times j$.

Todos los índices deben ser enteros mayores o iguales a cero. Para denominar una matriz se necesitan los dos índices.

matriz(i, j) donde:

- i es la cantidad de filas
- j , la cantidad de columnas
- matriz, el nombre de la matriz mediante el cual la computadora va a reconocer a ese conjunto de datos. Debe cumplir la misma normativa que una variable.

Los elementos de una matriz están organizados en filas y columnas como si fueran un grupo de vectores de los elementos cada uno o viceversa.

Los elementos de una fila varían todo, ellos igual valor para el primer índice, mientras que los de una columna tienen igual valor para el segundo, j .

El hecho de asignar el primer índice a las filas y el segundo a las columnas, es sólo una cuestión de nomenclatura. No hay ningún impedimento en asignar a las columnas y j a las filas, siempre y cuando seamos coherentes en todo el programa.

Ejemplo: la representación de una matriz mat de 4×5 , sería una matriz de cuatro filas y cinco columnas:

Matrices y soluciones de problemas - 1991

Matrices

Columnas

	1	2	3	4	5
1	mat(1,1)	mat(1,2)	mat(1,3)	mat(1,4)	mat(1,5)
2	mat(2,1)	mat(2,2)	mat(2,3)	mat(2,4)	mat(2,5)
3	mat(3,1)	mat(3,2)	mat(3,3)	mat(3,4)	mat(3,5)
4	mat(4,1)	mat(4,2)	mat(4,3)	mat(4,4)	mat(4,5)

El número de elementos de una matriz es $4 \times 5 = 20$
 Supongamos que tenemos una matriz nombres en la memoria de la computadora, con los nombres siguientes:

Columnas

	1	2	3	4	5
1	María				
2			Patricia		
3				Nicolás	
4		Isabella		Andrés	Elvira

1) Tal forma que:

nombres(1,1) = María
 nombres(2,3) = Patricia
 nombres(3,4) = Nicolás
 nombres(4,2) = Isabella
 nombres(4,4) = Andrés
 nombres(4,5) = Elvira

Cómo las matrices están almacenadas en la memoria de la computadora, sus elementos son de acceso directo

Algoritmos y estructuras de datos

Matrices

Podemos referirnos a una posición de memoria simplemente indicando sus coordenadas

El tipo de elementos que se almacenan en las matrices es el mismo en todos sus posiciones, todos pertenecen a todos a un mismo tipo, una única vez de ambos

Cuando se necesita almacenar información de distinto tipo se deben utilizar vectores paralelos

7.2 CARGA Y LECTURA DEL CONTENIDO DE UNA MATRIZ

Para cargar o leer una matriz se debe utilizar dos ciclos PARA en datos, recorriendo la matriz por filas y por columnas

Por ser ciclos anidados hay que indicar la variable i o j, según corresponda, sobre la sucesión PROXIMO
 PROXIMO i o PROXIMO j

Supongamos una matriz mat, de filas y columnas la forma de recorrerla es siguiente

• Ingreso de datos

```

PARA i = 1 A f
  PARA j = 1 A c
    INGRESAR 'ingresar dato', mat(i,j)
  PROXIMO j
PROXIMO i
    
```

Leerlo cargar en una matriz de $f \times c$ números enteros

```

PARA i = 1 A f
  PARA j = 1 A c
    PROXIMO j
    INGRESAR 'ingresar número', numero(i,j)
  PROXIMO j
PROXIMO i
    
```

Algoritmos y estructuras de datos

• **Inicialización**

```
PARA i = 1 A f
  PARA j = 1 A c
    PROXIMO i
      mat(i,j) = 0
    PROXIMO j
```

Ejemplo: poner a cero una matriz tiempo de 5 x 10

```
PARA i = 1 A 5
  PARA j = 1 A 10
    tiempo(i,j) = 0
  PROXIMO j
PROXIMO i
```

• **lectura para un cálculo**

```
PARA i = 1 A f
  PARA j = 1 A c
    suma = suma + mat(i,j)
  PROXIMO j
PROXIMO i
```

Ejemplo: realizar la suma de todos los elementos de una matriz sueldo de 8 x 15

```
PARA i = 1 A 8
  PARA j = 1 A 15
    suma = suma + sueldo(i,j)
  PROXIMO j
PROXIMO i
```

• **lectura para impresión**

```
PARA i = 1 A f
```

```
PARA i = 1 A c
  IMPRIMIR "Resultado: ", mat(i,j)
  PROXIMO i
PROXIMO j
```

Ejemplo: imprimir el contenido de una matriz notas de 40 x 9

```
PARA i = 1 A 40
  PARA j = 1 A 9
    IMPRIMIR "Nota ", nota(i,j)
  PROXIMO j
PROXIMO i
```

7.3 MÁXIMOS Y MÍNIMOS

El proceso de determinación de un valor máximo o mínimo de una matriz, es muy similar a la búsqueda en vectores. Se inicia la variable que contendrá el máximo o el mínimo, con el primer elemento de la matriz (1,1) luego se recorre nuevamente toda la matriz en la búsqueda de algún valor que supere o sea superado de acuerdo al tipo de valor que queremos encontrar: máximo o mínimo.

Cuando recorremos la matriz debemos hacerlo desde i = 1 hasta f y j = 1 hasta c. Esto significa volver a evaluar el elemento (1,1), pues a diferencia de la búsqueda en vectores no hacerlo traería como consecuencia no tener en cuenta el resto de la primera fila.

A continuación podemos ver la sintaxis de los algoritmos para buscar el máximo y el mínimo en una matriz

• **Búsqueda de un máximo**

```
max = mat(1,1)
PARA i = 1 A f
  PARA j = 1 A c
    Si mat(i,j) > max ENTONCES max = mat(i,j)
  PROXIMO j
PROXIMO i
```

Ejemplo buscar el peso máximo cargado en una matriz peso de 12×35 .

```
max = peso(1,1)
PARA i = 1 A 12
  PARA j = 1 A 35
    PROXIMO i
      SI peso(i,j) > max ENTONCES max = peso(i,j)
```

• Búsqueda de un mínimo

```
min = mat(1,1)
PARA i = 1 A 4
  PARA j = 1 A 7
    PROXIMO i
      SI mat(i,j) < min ENTONCES min = mat(i,j)
```

Ejemplo hallar la temperatura mínima cargada en una matriz temp de 4×7

```
min = temp(1,1)
PARA i = 1 A 4
  PARA j = 1 A 7
    PROXIMO i
      SI temp(i,j) < min ENTONCES min = temp(i,j)
```

7.4 ORDENAMIENTO DE MATRICES

Una matriz se ordena utilizando el método que para un vector. Al tener más de una columna debemos decidir por cuál de ellas queremos ordenarla.

También debemos tener en cuenta que cada vez que se produzca un intercambio de los elementos de una matriz, afecta a todos los elementos que estén situados en la misma fila.

Para intercambiar los elementos que están situados en la misma fila, pero en columnas distintas, el ser del mismo tipo (todos numéricos o todos

alfanuméricos) utilizaremos un ciclo que recorra todas las columnas para esa misma fila y los intercambie.

A continuación analizaremos el siguiente ejemplo.

Ordenar una matriz mat(0,4), por la primera columna.

```
COMENZAR
mat(0,4)
cota = 10
k = 1
FACTER MENIRAS k <= 0
  k = 0
  PARA i = 1 A cota - 1
    SI mat(i,1) > mat(k + 1,1) ENTONCES
      PARA j = i A 4
        aux = mat(i,j)
        mat(i,j) = mat(k + 1,j)
        mat(k + 1,j) = aux
      PROXIMO j
      k = i
    FIN SI
  PROXIMO i
  cota = k
REPETIR
IN
```

Para recorrer la matriz se necesitan dos ciclos (PARA, uno para comparar los elementos en la columna i mat(i,j) > mat(k + 1,j), y otro para intercambiar los elementos de todas las columnas, cuando está desordenada la columna i PARA j = i A 4

7.5 EJERCICIOS DE APLICACIÓN

Ejercicio 1

Generar una matriz mat de 5×10 , introduciendo los valores por teclado.

Se pide imprimir.

- a) la raíz.
- b) la suma de los elementos impares
- c) El elemento $mat[3,5]$

COMIENZO

$mat[5,10]$
 $suma = 0$

PARA $i = 1$ A 5

PARA $j = 1$ A 10

INGRESAR "Ingresar dato ", $mat[i,j]$

PROXIMO i

PARA $i = 1$ A 5

PARA $j = 1$ A 10

$S = (1) \wedge mat[i,j] < 0$ ENTONCES $suma = suma + mat[i,j]$

PROXIMO j

PARA $i = 1$ A 5

PARA $j = 1$ A 10

IMPR MIR " Elemento ", " ", " ", " ", $mat[i,j]$

PROXIMO j

IMPR MIR " La suma de elementos "pares es ", suma

IMPR MIR "El elemento $mat[3,5]$ es igual a ", $mat[3,5]$

FIN

En el ejercicio se dimensionó a matriz y se colocó a cero el acumulador utilizado para la suma de los elementos impares. Luego se carga la matriz con los 50 valores

La determinación de si un elemento es impar se llevó a cabo elevando a 1 a el orden de dicho elemento, si el resultado es menor que cero, el elemento es impar.

Si $(1) \wedge mat[i,j] < 0$, $mat[i,j]$ es impar

Podemos observar que cuando queremos imprimir un elemento en particular, accedemos directamente a él colocándolo el valor de los índices en esa posición

IMPR MIR "El elemento $mat[3,5]$ es igual a ", $mat[3,5]$

Ejercicio 2

Cargar una matriz $mat[5,5]$, introduciendo los valores por teclado.
Se pide imprimir

- a) la suma de los elementos de la diagonal principal
- b) la suma de todos los elementos
- c) la suma de la columna 5

COMIENZO

INICIO

CARGA

DIAGONAL

TOTAL

COLUMA 5

MPRESION

FIN

INICIO

$mat[5,5]$

$sumdiag = 0$

$sumtot = 0$

$sumcol5 = 0$

RETORNAR

CARGA

PARA $i = 1$ A 5

PARA $j = 1$ A 5

INGRESAR "Ingresar dato ", $mat[i,j]$

PROXIMO j

```

PROXIMO:
RETORNAR

DAGONAL.
PARA i = 1 A 5
    PARA j = 1 A 5
        SI i = j ENTONCES sumdiag = sumdiag + mat(i,j)
    PROXIMO j
PROXIMO i
RETORNAR

TOTAL:
PARA i = 1 A 5
    PARA j = 1 A 5
        sumatof = sumatof + mat(i,j)
    PROXIMO j
PROXIMO i
RETORNAR

COLUM 5.
PARA i = 1 A 5
    sumacol5 = sumacol5 + mat(i,5)
PROXIMO i
RETORNAR

IMPRESION
MAPRIMR "La suma de la diagonal principal es ", sumdiag
MAPRIMR "La suma de todos los elementos es ", sumatof
MAPRIMR "La suma de la columna 5 es ", sumacol5
RETORNAR

```

En este caso se le dimos el programa en 6 subprogramas.

Se trabajó con 3 arrays adores de suma la determinación de la suma de la diagonal se basó en considerar para la suma que los elementos que tienen iguales ambos índices y se puede apreciar como se suma la columna 3 a partir de ocurrir el ciclo PARA que recorre sólo los elementos de la columna 5. Esto se logra manteniendo constante el valor de la columna a recorrer mat(i,5)

100 - Algoritmos y Programación de Datos

Ejercicio 3

Generar una matriz de n filas y m columnas. Cargar la matriz con números enteros mayores a cero. Se pide calcular e imprimir:

- El valor máximo de toda la matriz
- El valor mínimo de toda la matriz
- La cantidad de veces que se repite el número 10

```

COMENZO
NICIO
CARGA
PARA i = 1 A n
    PARA j = 1 A m
        INGRESAR "Ingrese la cantidad de filas ", n
        INGRESAR "Ingrese la cantidad de columnas ", m
        mat(n,m)
    RETORNAR

CARGA
PARA i = 1 A n
    PARA j = 1 A m
        INGRESAR "Ingrese un número.", mat(i,j)
PROXIMO j
RETORNAR

MAXIMO.
max = mat(1,1)
PARA i = 1 A n
    PARA j = 1 A m
        SI mat(i,j) > max ENTONCES max = mat(i,j)

```

100 - Algoritmos y Programación de Datos

PROXIMO I
RETORNAR

MINIMO

func mat(1,1)

PARA - 1 A n

PARA - 1 A n

if mat(i,j) > min ENTONCES min = mat(i,j)

PROXIMO I

RETORNAR

VECES: 10

contador = 0

PARA i = 1 A n

PARA - 1 A n

if mat(i,j) > max ENTONCES contador = contador + 1

PROXIMO I

RETORNAR

IMPRESION

IMPRIMIR "El valor máximo es ", max

IMPRIMIR "El valor mínimo es ", min

IMPRIMIR "Cantidad de repeticiones del número 0 ", contador

RETORNAR

El programa de ejercicio esta formado por 6 subprogramas

La cantidad de elementos de la matriz es variable, pues n y m pueden tomar distintos valores para ejecuciones diferentes

Los algoritmos de cálculo, MAXIMO, MINIMO y VECES = 0, por si mismos o otros analizados anteriormente. Podemos apreciar en cada recorrido realizado a la matriz, como los ciclos PARA están de limitados en la cantidad de repeticiones, por los valores asignados a n y m

Algoritmos y estructuras de datos

Ejercicio 4

Crear una matriz de 25 filas y 5 columnas con las notas obtenidas por 25 alumnos durante el primer semestre del año

La columna 1 se carga con el número de alumno (del 1 al 25)

Las columnas 3, 4 y 5 deben completarse con las calificaciones en las tres asignaturas cursadas: Matemática, Física y Química. La columna 2 se carga con el promedio de las tres notas

Se pide mostrar:

- La matriz ordenada por promedio en forma descendente.
- La nota promedio general de Matemática, para los 25 alumnos

COMENZO

INICIO

CARGA

ORDEN

PROMEDIO

APRESION

FIN

INICIO

notas(25,5)

RETORNAR

CARGA

PARA i = 1 A 25

notas(i,1) = i

INGRESAR "Ingrese la nota de Matemática ", nota(i,3)

INGRESAR "Ingrese la nota de Física ", nota(i,4)

INGRESAR "Ingrese la nota de Química ", nota(i,5)

nota(i,2) = (nota(i,3) + nota(i,4) + nota(i,5)) / 3

PROXIMO

RETORNAR

ORDEN:

ma(25,5)

cola = 25

k = 1

Algoritmos y estructuras de datos

Matrices

```

FACTR MENTRAS k <= 0
  k = 0
  PARA i = 1 A tota - 1
    SI mat(i,2) < mat(i + 1,2) ENTONCES
      PARA j = 1 A 5
        aux = mat(i,j)
        mat(i,j) = mat(i + 1,j)
        mat(i + 1,j) = aux
      PROXIMO
    FIN SI
  PROXIMO
  tota = k
  REPETIR
  RETORNAR k

PROMEDIO
si na = 0
  PARA i = 1 A 25
    suma = suma + nota(i,3)
  PROXIMO
  promat = suma / 25
  RETORNAR

IMPRESION:
'MPRM R 'Matriz o decada por promedio'
PARA i = 1 A 25
  MPRMIR 'Numero de alumno', nota(i,1)
  MPRMIR 'Promedio', nota(i,2)
  MPRMIR 'Materia:', nota(i,3)
  MPRMIR 'Escala', nota ,41
  MPRMIR 'Química', nota(i,5)
PROXIMO
IMPRMIR 'Promedio general de Matemática', promat
RETORNAR
  
```

El programa está compuesto por 5 subprogramas: INICIO, CARGA, el subprograma de ordenamiento ORDEN, un cálculo del promedio de una

1000 EJERCICIOS Y SUBPROGRAMAS DE MATRICES

Matrices

se utiliza en particular PROMEDIO y el subprograma de impresión también llamado IMPRESION

En el subprograma de carga se utilizó un solo ciclo pues los valores se cargaron columna a columna. Lo mismo sucedió al calcular el promedio general de Matemática

El ordenamiento responde al ciclo forzado en el punto anterior, con la diferencia que ahora se ordena tomando como criterio la columna 2 que contiene los promedios.

Ejercicio 5

a) Crear una matriz de 10 filas y 15 columnas. Los primeros 9 filas se cargan con valores ingresados externamente. La fila 10 debe correr la sumatoria de las 15 columnas para las 9 filas restantes. Se pide imprimir:

- El valor máximo de la fila 10
- La cantidad de ceros de la columna 15

```

COMIENZO
INICIO
CARGA
MAXIMO
CEROS
IMPRESION
FIN

INICIO.
mat(10,15)
PARA i = 1 A 15
  mat(10,i) = 0
PROXIMO
KLICKNAR

CARGA.
PARA i = 1 A 9
  PARA j = 1 A 15
    INGRESAR "Ingrese un número, mat(i,j)"
  
```

1000 EJERCICIOS Y SUBPROGRAMAS DE MATRICES

Matrices

$mat[0, j] = mat[0, j] + mat[i, j]$

PROXIMO

PROXIMO
RETURN

MAXIMO

$max = mat[0, j]$

PARA $j = 2$ A 15

SI $mat[10, j] > max$ ENTONCES $max = mat[10, j]$

PROXIMO
RETURN

CONTADOR

CONTADOR = 0

PARA $i = 1$ A 10

SI $mat[i, 15] = 0$ ENTONCES CONTADOR = CONTADOR + 1

PROXIMO
RETURN

IMPRESION

PRINT "El valor máximo de la fila 10 es: " max

PRINT "La cantidad de ceros de la columna 15 es: " contador

RETURN

El programa está dividido en 5 subprogramas: INICIO, CARGA, MAXIMO, CERO y IMPRESION

La cuenta la particularidad que a fila 0 se obtiene como suatoria de las y filas restantes. Podríamos decir que a fila 0 no es otra cosa que un contador de 15 acumuladores. Le que se acumula sobre 9 elementos que corresponden a cada una de las 9 primarias filas.

De la misma manera, tenemos el contador que cuenta la cantidad de ceros de la columna 15. Veremos como para recorrerla tenemos que fijar el índice en 15.

Matrices

RESUMEN

Una matriz es un conjunto de datos homogéneos organizados bajo un mismo nombre, cada uno de los cuales debe referenciarse por dos índices, i y j también llamados subíndices, que identifican las filas y las columnas de la matriz $i \times j$.

Ambos índices deben ser enteros, mayores o iguales a cero. Para dimensionar una matriz se necesitan los dos índices.

matriz (i, j) donde:

- i es la cantidad de filas.
- j es la cantidad de columnas.

matriz, el nombre de la matriz mediante el cual la computadora reconoce a ese conjunto de datos. Debe cumplir la misma nomenclatura que una variable.

Los elementos de una matriz están organizados en filas y columnas como si fueran un grupo de vectores de elementos cada uno o viceversa.

Los elementos de una fila tienen todos el mismo valor para el primer índice, i , mientras que los de una columna tienen igual valor para el segundo índice, j . El hecho de que los de una columna tienen igual valor para el segundo índice, es solo una cuestión de nomenclatura. No hay ningún impedimento para que las columnas y las filas, siempre y cuando sean coherentes, en todo el programa.

El tipo de elementos que se almacenan en las matrices es el mismo en toda, sus posiciones, todos numéricos o todos alfanuméricos, pero nunca una mezcla de ambos.

• Ingreso de datos

PARA $i = 1$ A i

PARA $j = 1$ A j

PROXIMO
INGRESAR "Ingresar dato: ", mat[i, j]

Matrices

PROXIMO i

• Inicialización

```
PARA i = 1 A n
  PARA j = 1 A c
    mat(i,j) = 0
  PROXIMO j
PROXIMO i
```

• Lectura para un cálculo

```
PARA i = 1 A f
  PARA j = 1 A c
    suma = suma + mat(i,j)
  PROXIMO j
PROXIMO i
```

• Lectura para impresión

```
PARA i = 1 A f
  PARA j = 1 A c
    IMPRIMIR "Resultado:", mat(i,j)
  PROXIMO j
PROXIMO i
```

• Búsqueda de un máximo

```
max = mat(1,1)
PARA i = 1 A f
  PARA j = 1 A c
    Si mat(i,j) > max ENTONCES max = mat(i,j)
  PROXIMO j
PROXIMO i
```

• Búsqueda de un mínimo

```
min = mat(1,1)
PARA i = 1 A f
```

Matrices

```
PARA i = 1 A c
  Si mat(i,1) > min ENTONCES min = mat(i,1)
PROXIMO i
```

• Ordenamiento

```
mat(n,m)
cola = n
k = 1
HACER MIENTRAS k <= 0
  < - 0
  PARA i = 1 A cola - 1
    Si mat(i,1) > mat(i+1,1) ENTONCES
      PARA j = 1 A m
        aux = mat(i,j)
        mat(i,j) = mat(i+1,j)
        mat(i+1,j) = aux
      PROXIMO j
      k = i
    FIN SI
  PROXIMO i
  cola = k
REPETIR
```



Ejercicios Propuestos

1. Crear una matriz $m \times n$ de n filas y m columnas, cuyos elementos sean múltiplos de m . Se pide imprimir:

- a) la matriz completa
- b) los elementos de la fila del medio

2. Crear una matriz de $m \times n$ de 7 filas y 9 columnas.

- a) la suma de las columnas pares
- b) la suma de las filas impares.

3. Crear una matriz $m \times n$ de 8 filas y 12 columnas en la que los elementos de las filas pares sean múltiplos de 2 y los de las filas impares sean 5 por acción de la columna.

A partir de ella crear otra matriz $m \times n$ cuyas columnas impares sean los de $m \times n$ elevadas al cuadrado y las pares as de $m \times n$.

4. Construir una matriz $m \times n$ de 5 filas y 5 columnas de tal forma que los elementos de la diagonal sean unos y los demás elementos se ingresen desde afuera imprimiendo.

5. Construir una matriz $m \times n$ de 15 filas y 15 columnas de tal forma que los elementos de la diagonal sean unos y los demás elementos sean ceros. Imprimir.

6. Construir una matriz de 0 filas y 9 columnas, de tal forma que el valor de los elementos de cada columna sean, respectivamente, los totales de multiplicar del 1 a 9.

7. Cargar una matriz de 12 filas y 31 columnas con los datos que se piden.

170 - EJERCICIOS Y ACTIVIDADES DE ALGORITMOS Y ESTRUCTURAS DE DATOS

ten los meses del año y las columnas la temperatura media de cada día. Se pide calcular e imprimir:

- a) la temperatura media de agosto
- b) El día más cálido del año.
- c) El día más frío del año.
- d) El día más frío de julio.
- e) El día más cálido de enero.

171 - EJERCICIOS Y ACTIVIDADES DE ALGORITMOS Y ESTRUCTURAS DE DATOS

Ejercicios Combinados

8.1 INTRODUCCIÓN

A lo largo del texto se han analizado los distintos tipos de estructuras, ya sea desde el punto de vista de los tipos de instrucciones, como desde el punto de vista de organización de los datos.

Como corolario del desarrollo de este curso de introducción a la programación, a partir de algoritmos escritos en pseudocódigo, y de la construcción y manipulación de estructuras internas como los vectores y las matrices, vamos a desarrollar en este capítulo un conjunto de ejercicios que combinen el manejo de vectores y matrices.

lógicamente que esto nos obliga a integrar los conceptos estudiados en todos los unidades temáticas vistas hasta ahora, lo que desde la perspectiva del aprendizaje significativo, es alcanzar por el estudiante, es decir, un resultado positivo.

8.2 EJERCICIOS RESUELTOS

8.2.1 Ejercicios con carga con ciclo PARA...PROXIMO

Ejercicio 1

Una empresa de productos alimenticios posee 30 empleados. Los empleados concurren diariamente a trabajar en el horario de 9 a 18 hrs. Mensualmente cobijan un sueldo y dependido de las horas trabajadas.

IAFS.

[illegible]

Ejercicios Combinados

Se tienen cargados en un vector llamado nombre\$ los nombres de los 30 empleados

Se pide calcular e imprimir

- Salario anual de cada empleado
- Total de sueldos pagados cada mes
- Máximo sueldo pagado en cada mes
- Porcentaje que representa cada sueldo anual sobre el total
- Nombre del empleado que haya cobrado el mínimo sueldo en el primer semestre de año
- Cantidad de meses en los que se haya pagado menos de 45 000\$ en total en concepto de sueldos

Se pide además ordenar los sueldos anuales en forma ascendente e imprimirlos de la siguiente forma

```
NOMBRE      SUELDO ANUAL
COMINFN/CO
INCIO
CARGA
SUELDO ANUAL
SUELDO MENSUAL
PORCENTAJE
MINIMO SEMESTRAL
MENOS 45000
ORDEN
MPRESION
FIN
```

El programa se dividirá en 9 subprogramas

```
INCIO
sdo[12,30]
sae[30].nom[30] por[0] nomord[30]
sm[2] max[12]
PARA i = 1 A 30
  INGRESAR "Nombre:", nom[i]
  nomord[i] = nom[i]
```

Ejercicios Combinados

PROXIMO
RETORNAR

Se utilizará una matriz de sueldos sd[12,30], es 12 filas para los meses de año y 30 columnas para cada uno de los empleados. En ella se cargarán los sueldos que figuran en la planilla

Se trabajará con 6 vectores, 4 de 30 elementos y 2 de 12 elementos

```
sae[30] sueldo anual de cada empleado
nom[30] nombre de cada empleado
nom[30] nombre de cada empleado
sm[12] total de sueldos pagados cada mes
rmax[12] máximo sueldo pagado en cada mes
por[30] porcentaje que representa cada sueldo anual sobre el total
```

```
CARGA
PARA i = 1 A 12
  INGRESAR "Mes ", mes
  PARA j = 1 A 30
    INGRESAR "Número de empleado ", j
    INGRESAR "Cantidad de horas ", horas
    INGRESAR "Valor de la hora.", v[
      sdo[mes,pro] horas * v[
  PROXIMO
PROXIMO
RETORNAR
```

Luego del primer ciclo PARA se ingresa el mes por el que depende la carga del orden de las planillas la matriz sueldo se carga con el producto entre los horas trabajados y el valor de la hora

```
SUELDOANUAL
PARA i = 1 A 30
  PARA j = 1 A 12
    sae[i] = sae[i] + sdo[j,i]
  PROXIMO
PROXIMO
RETORNAR
```

Ejercicios Combinados

El vector `sac()` acumula a suma de todos los sueldos por cada dos por el empleado durante los 12 meses. Al ejecutarse primero el ciclo 1 estamos recorriendo a matriz por columna o sea por empleado

```
SUELDO MENSUAL
PARA i = 1 A 12
  PARA j = 1 A 30
    sm(j) = sm(j) + sdo(i,j)
  PROXIMO j
PROXIMO i
RETORNAR
```

El vector `sm()` acumula los sueldos por mes por los 30 empleados. En este caso la matriz se recorre por filas

```
MAX MOM NSUAL
PARA i = 1 A 12
  max(i) = sdo(i,1)
  PARA j = 2 A 30
    Si sdo(i,j) > max(i) ENTONCES max(i) = sdo(i,j)
  PROXIMO j
PROXIMO i
RETORNAR
```

Debemos calcular 12 próximos, por eso el primer ciclo PARA indica esta repetición del cálculo. Luego para cada mes inicializamos el vector de máximo con el sueldo de primer empleado `max(i) = sdo(i,1)`, y lo comparamos con los 29 restantes, arancando el segundo ciclo de 2: PARA j = 2 A 30

```
PROCESAR
sdo = 0
PARA i = 1 A 30
  sdo = sdo + sac(i)
PROXIMO i
PARA j = 1 A 30
  par(j) = sac(j) / sdo * 100
PROXIMO j
RETORNAR
```

Algoritmos y Estrategias de Programación

Ejercicios Combinados

Los 30 porcentajes se obtienen al dividir el sueldo anual de cada empleado `sac()` por el total pagado en el año a todos los empleados. Esta suma se acumula en la variable `sdo`

```
MINIMO SEMESTRAL
msem = sdo(1,1)
PARA i = 1 A 6
  PARA j = 1 A 30
    Si sdo(i,j) < msem ENTONCES msem = sdo(i,j)
  PROXIMO j
PROXIMO i
RETORNAR
```

El vector semestral se determinó recorriendo la matriz hasta a fila 6 luego a junio.

```
MENOS 45000
cm = 0
PARA i = 1 A 12
  Si sm(i) < 45000 ENTONCES cm = cm + 1
PROXIMO i
RETORNAR
```

El contador `cm`, lleva la cuenta de la cantidad de meses que se pagó menos de 45000\$, en concepto de sueldos.

ORDEN:

```
cola = 30
l = 1
HACER MIENTRAS k <= 0
  k = 0
  PARA i = 1 A cola
    Si sac(i) > sac(i+1) ENTONCES
      aux = sac(i)
      sac(i) = sac(i+1)
      sac(i+1) = aux
      aux$ = record$(i)
```

Algoritmos y Estrategias de Programación


```

PARA I = 1 A 10
  IMPR MIR "Número de proyecto",
  IMPR MIR "etapas trabajadas", h(i)
PROX MO
IMPR MIR "Sueldos mensuales"
PARA I = 1 A 15
  IMPR MIR "Nombre:", nom$(I), "Sueldo:", sdo(I), "$"
  NEXT MO
IMPR MIR "Aires consumidos máximo"
PARA I = 1 A 15
  SI sdo(I) > sdo(15) ENTONCES IMPR MIR "nom$(I)"
PROX MO
RETORNAR

```

8.2.2 Ejercicios con carga con ciclo HACER MIENTRAS...REPETIR

Ejercicio 3

Una empresa se dedica a la comercialización de combustibles. Cuenta con 5 estaciones de servicio distribuidas en Capital y Gran Buenos Aires. Los nombres de las estaciones se deberán cargar en un vector en modo

a) En cada estación de servicio se venden los siguientes productos

- 1 Nafta común
- 2 Nafta especial
- 3 Nafta sin plomo
- 4 Gas Oil

Muestra dentro cada estación de servicio a cuánto, las comprobantes de las ventas en pesos con los siguientes datos:

Número de estación
Codigo de combustible
Importe de la venta

Se pide calcular e imprimir:

1. Total recaudado por cada estación

- a) Total recaudado por cada estación
- b) Total recaudado por cada tipo de combustible
- c) Nombre del combustible que recaudó menos, para cada estación
- d) Máxima recaudación de cada combustible
- e) Estación de mayor recaudación
- f) Cantidad de estaciones que hayan facturado más de 15000\$ en total para los 4 combustibles
- g) Porcentaje de facturación y nombre de combustible que facturó más

```

COMIENZO
INICIO
CARGA
REC ESTACION
REC COMBUSTIBLE
MIN ESTACION
MAX COMBUSTIBLE
MAX ESTACION
FACTURACION 15000
PORCENTAJE
FIN

```

El programa se divide en 9 subprogramas

```

INICIO
  venta(1,5,4)
  rc(1,5) := rc(1,5) + nc$(1,5)
  rc(4) := rc(4) + maxc(4) + po(4)
  nc$(1) = "Nafta Común"
  nc$(2) = "Nafta Especial"
  rc$(3) = "Nafta sin Plomo"
  rc$(4) = "Gas Oil"
  PARA I = 1 A 15
    NGR SAR "Nombre de estación:", nc$(i)
  PROXIMO
  RETORNAR

```

Se creó la matriz venta, que almacenará las ventas de los 4 combustibles

1. Total recaudado por cada estación

FUNCIONES COMBINADAS

para las 15 estaciones

Se dimensiona / vector:

re[15] "recaudación por estación"

re[i] = 5) "menor recaudación de un combustible por estación"

re[i] = 15) "nombre de cada estación"

re[i] = 4) "recaudación por combustible"

re[i] = 4) "nombre de cada combustible"

re[i] = 4) "máxima recaudación de cada combustible"

re[i] = 4) "porcentaje de facturación de cada combustible"

CARGA

INGRESAR "Número de estación" est

HACER MENORES est < 0

INGRESAR "Código de combustible", cod

INGRESAR "Venta en pesos", ven

ventas[i, cod] = ven

INGRESAR "Número de estación", est

REPETIR

RETORNAR

A diferencia de los ejemplos de carga vistos en los ejercicios anteriores, ahora desconocemos la cantidad de comprobantes remitidos por todas las estaciones. Recurrimos a utilizar uno de los variables como "fin de archivo", en este caso el número de estación. Cuando no queden más comprobantes por procesar ingresaremos un cero que indicará el fin de la carga de datos

REC ESTACION.

PARA i = 1 A 15

PARA j = 1 A 4

re[i] = re[i] + vent[i, j]

PROXIMO i

PROXIMO j

RETORNAR

La "recaudación" por estación se calcula recorriendo la matriz por filas

REC COMBUSTIBLE.

PARA i = 1 A 4

PARA j = 1 A 15

180. Algoritmos y estructuras de datos

FUNCIONES COMBINADAS

re[i] = re[i] + vent[i, j]

PROXIMO i

PROXIMO j

RETORNAR

En este caso se debe recorrer por columnas para determinar la "recaudación por combustible"

MAXIMIZACION

PARA i = 1 A 15

re[i] = vent[i, 1]

PARA j = 2 A 15

Si vent[i, j] < re[i] ENTONCES re[i] = vent[i, j]

PROXIMO j

PROXIMO i

RETORNAR

Se obtuvieron 15 "máximos" uno para cada estación, referidos al "combustible de menor recaudación"

Se comienza suponiendo para cada estación que "recaudación" es la "menor" recaudación: re[i] = vent[i, 1], y se compara su valor con los "restantes" PARA j = 2 A 15

MAXIMIZACION

PARA i = 1 A 4

max[i] = vent[i, 1]

PARA j = 2 A 15

Si vent[i, j] > max[i] ENTONCES max[i] = vent[i, j]

PROXIMO j

PROXIMO i

RETORNAR

Se hallaron 4 "máximos" entre las "5" estaciones, uno para cada combustible

Se comienza asignando para cada combustible, a la estación 1 como "máximo" max[i] = vent[i, 1] y luego se compara con las 14 restantes.

180. Algoritmos y estructuras de datos - 187

Funciones Combinadas

```

MAXESTACION
    cmax = rc(i)
    PARA i = 2 A 5
        SI rc(i) > cmax ENTONCES cmax = rc(i)
    PROXIMO
RETORNAR
    
```

En este subprograma se calcula la máxima recaudación entre las 15 estaciones. Al ser un valor único, se a maximiza esta recaudación en términos, para luego seleccionar el subprograma de "impresión", las estaciones que respo-

```

FACTURACION-15000.
    c = 5
    PARA i = 1 A 5
        SI rc(i) > 15000 ENTONCES c = rc(i) + 1
    PROXIMO
RETORNAR
    
```

La variable c es un contador que almacena la cantidad de estaciones que facturaron mas de \$5000\$. Se utiliza como un contador el vector rc(i), pues contiene las recaudaciones de todas las estaciones

```

PORCENTAJE:
    f = 0
    PARA i = 1 A 4
        f = f + rc(i)
    PROXIMO
    PARA j = 1 A 4
        for(j) = rc(j) / f * 100
    PROXIMO
    pmax = for(1)
    PARA i = 2 A 4
        SI for(i) > pmax ENTONCES pmax = for(i)
    PROXIMO
RETORNAR
    
```

El porcentaje de facturación de cada combustible está determinado por la expresión: $por(i) = rc(i) / f * 100$, donde f es el acumulador de recaudación

100. Funciones y Combinadas de Datos

Funciones Combinadas

res donde se almacenará el total recaudado por todos los combustibles y rc(i) la recaudación individual de cada uno de los 4 combustibles

```

IMPRESION
    IMPRIMIR "Recaudación por estación"
    PARA i = 1 A 15
        IMPRIMIR "Estación ", i, "Recaudación ", rc(i), "$"
    PROXIMO
    IMPRIMIR "Recaudación por combustible"
    PARA i = 1 A 4
        IMPRIMIR "Combustible ", rc(i), "Recaudación ", rc(i), "$"
    PROXIMO
    IMPRIMIR "Combustible de mayor recaudación por estación"
    PARA i = 1 A 15
        IMPRIMIR "Estación número ", i
    PROXIMO
    SI ve("a", i) = mc(i) ENTONCES IMPRIMIR mc(i)
    PROXIMO
    IMPRIMIR "La recaudación máxima por combustible"
    PARA i = 1 A 4
        IMPRIMIR "Combustible ", mc(i), "Recaudación máxima ", max(i), "$"
    PROXIMO
    IMPRIMIR "Estación de máxima recaudación"
    PARA i = 1 A 5
        SI rc(i) = cmax ENTONCES IMPRIMIR mc(i)
    PROXIMO
    IMPRIMIR "Estaciones con facturación mayor a $5000"
    PARA i = 1 A 4
        IMPRIMIR "Porcentaje de facturación de cada combustible"
    PROXIMO
    IMPRIMIR "Combustible de mayor porcentaje"
    PARA j = 1 A 4
        SI for(j) > pmax ENTONCES IMPRIMIR for(j)
    PROXIMO
RETORNAR
    
```

100. Funciones y Combinadas de Datos

Ejercicios combinados

```
ac(i,q) alumnos por curso
rc(i,q): recaudación por curso
pc(i,q) precio por curso.
```

CARGA

PARA i = 1 A 15

INGRESAR "Número de vendedor", nrov

PARA j = 1 A 10

INGRESAR "Cantidad de alumnos", ca

INGRESAR "Cantidad de cursos", cc

ca = ca * nrov, nroc = ca

recau(i,nrov,nroc) = ca * pc(i)

PROXIMO j

PROXIMO i

RETORNAR

Se cargarán los datos en forma paralela a la cantidad de alumnos
ca = nrov(nrov,nroc) ca, y la recaudación recau(nrov,nroc) cc * pc(i)

ALUMNOS CURSO

PARA i = 1 A 10

PARA j = 1 A 15

ac(i) = ac(i) + recau(i,j)

PROXIMO j

PROXIMO i

RETORNAR

Se recorrió la matriz por columnas para los 15 cursos

ALUMNOS VENDEDOR

PARA i = 1 A 15

PARA j = 1 A 10

ac(i) = ac(i) + recau(i,j)

PROXIMO j

PROXIMO i

RETORNAR

Ahora se recorrió la matriz por filas para los 15 vendedores,
CURSO MAS INSCRIPTOS

Ejercicios combinados

```
instrax = cc(i)
```

PARA i = 1 A 10

Si ac(i) > instrax ENTONCES instrax = ac(i)

PROXIMO

RETORNAR

La variable instrax, almacena la mayor inscrición para un curso. En la
expresión se determina el nombre del curso que responde a esa inscrip-
ción.

MAX VENDEDOR

PARA i = 1 A 15

maxv(i) = ca i(nv,i)

PARA j = 1 A 10

Si ca i(nv,i) > maxv(i) ENTONCES maxv(i) = ca i(nv,i)

PROXIMO

PROXIMO j

RETORNAR

Para cada vendedor hay que calcular la máxima cantidad de inscriptos,
valor que se almacena en maxv(i). Este vector se recorre 15 veces por la
cantidad de cursos del primer curso para luego compararlo con los 9 res-
tantes

REC VENDEDOR

PARA i = 1 A 15

PARA j = 1 A 10

rv(i) = rv(i) + recau(i,j)

PROXIMO

PROXIMO j

RETORNAR

El vector rv(i) acumula la recaudación total por vendedor, para esto se
recorre la matriz por filas

REC CURSO

PARA i = 1 A 10

PARA j = 1 A 15

rc(i) = rc(i) + recau(i,j)

1. Ejercicios (continúa)

```
PROXIMO:
PROXIMO
RETORNAR
```

El vector `rec()` acumula la recaudación total por vendedor, ahora se recorre la matriz por columnas

```
MAX UNA NSCRIPCION
max = cantul[1,1]
PARA i = 1 A 15
    PARA j = 1 A 10
        Si cantul[i,j] > max EN ONCES max = cantul[i,j]
    PROXIMO
PROXIMO
RETORNAR
```

El máximo calculado en este caso es de toda la matriz, pues es por una sola inscripción

```
VENDIDOR 5
c = 0
PARA i = 1 A 10
    Si cantul[5,i] = 0 ENTONCES c = c + 1
PROXIMO i
RETORNAR
```

La variable `c` es un contador que cuenta la cantidad de cursos que no realizó inscripciones el vendedor 5. El valor de las filas en la matriz queda fijo por recorrer exclusivamente esa fila (cantul[5,]).

```
IMPRESION
IMPRIMIR "Alumnos por curso"
PARA i = 1 A 10
    IMPRIMIR "Curso ", nc$(i)
    ALUMNOS "Alumnos por vendedor"
    PARA j = 1 A 15
        IMPRIMIR "Vendedor ", nv$(j), "Alumnos:", av(j)
    PROXIMO j
PROXIMO i
```

2. Ejercicios (continúa)

```
IMPRIMIR "Curso de máxima inscripción"
PARA i = 1 A 10
    Si ac(i) = max {ENTONCES IMPRIMIR nc$(i)}
PROXIMO i
IMPRIMIR "Máxima inscripción por vendedor"
PARA i = 1 A 15
    IMPRIMIR "Vendedor ", nv$(i)
    IMPRIMIR "Máxima inscripción ", maxv(i)
    IMPRIMIR "Curso"
    PARA j = 1 A 10
        Si maxv(i) = cantul[i,j] ENTONCES IMPRIMIR nc$(j)
    PROXIMO j
PROXIMO i
IMPRIMIR "Recaudación por vendedor"
PARA i = 1 A 15
    IMPRIMIR "Vendedor ", nv$(i)
    IMPRIMIR "Recaudación por curso"
    PARA j = 1 A 10
        IMPRIMIR "Vendedor de máxima inscripción ", nv$(i)
        PARA k = 1 A 15
            PARA l = 1 A 10
                Si maxv(i) = cantul[i,l] ENTONCES IMPRIMIR nc$(l)
            PROXIMO l
        PROXIMO k
    PROXIMO j
    IMPRIMIR "Recaudación ", rec(i)
PROXIMO i
IMPRIMIR "Vendedor de máxima inscripción"
PARA i = 1 A 15
    k = 0
    PARA j = 1 A 10
        Si cantul[i,j] = 0 ENTONCES k = k + 1
    PROXIMO j
    Si k = 1 ENTONCES IMPRIMIR nv$(i)
PROXIMO i
IMPRIMIR "El vendedor ", nv$(5), "no realizó inscripciones en ", c5 "cursos"
RETORNAR
```

La impresión presenta todos los vendedores calculados hasta ahora para mostrar resultados. Nueva mente se recorre la matriz de un switch, esta vez para encontrar la inscripción de verdadero de máxima inscripción (para el curso), pues puede darse el caso de que el mismo vendedor tenga logrado más de una vez la máxima inscripción y podría imprimirse dos o más veces su nombre.

Ejercicio 5

a) empresa tiene 17 centros de embotellado de sus gaseosas. En cada centro habitan en promedio 120 personas.

b) Normalmente se procede a embotellar agua, tereno como mínimo que tener 1000 envases y como máximo 15000.

Las gaseosas que se emvasan son:

1. COLA
2. LIMON
3. NARANJA
4. POMERO

c) Costo de embotellar agua varía para cada centro y cada gaseosa.

d) Costo de emvasado de cada envase, de cualquier gaseosa es de 0.10\$.

e) Costo de mano de obra es de 0.05\$ por envase.

f) Para los centros 1, 2, 3 y 6 hay que considerar que los costos se reducen en un 25% debido a la ubicación cercana de dichos centros.

g) A costo de cada litro, se le carga el 100% para cada una de las personas.

h) Cada centro entrega a costo de 20 días de trabajo, una planilla con la cantidad de envases llenados en ese mes para cada gaseosa.

NUMERO DEL CENTRO

NOMBRE DEL CENTRO

NUMERO DE GASEOSA	COSTO DE EMBOTELLADO	CANTIDAD ENVASADA
	[por litro]	[centros]

Se pide calcular e imprimir:

a) Cantidad emvasada por centro.

b) Cantidad emvasada de cada gaseosa.

c) Costo de embotellar agua de cada centro.

d) Para cada gaseosa calculamos el número del centro en el que se emvasa con mayor cantidad de litros.

e) Porcentaje que representa el costo de emvasado de cada centro sobre el costo total de la empresa.

f) Si consideramos que se vende todo lo que se produce, ¿Cuál fue la facturación total de cada centro?

g) Nombre de la gaseosa de mayor facturación por los 17 centros.

Imprimir los items a, c y f ordenados en forma ascendente por facturación.

COMIENZO
INICIO
CARGA
CANTIDAD CENTRO
CANTIDAD GASEOSA
COSTO CENTRO
MAXIMO GASEOSA
PORCENTAJE
FACTURACION CENTRO
MAXIMO CENTRO
ORDEN
IMPRESION
FIN

Vamos a trabajar con 11 subprogramas.

1. INICIO.
costo(1/4) : costo(1/4)
nc\$(1/4) : nc\$(1/4)
ng\$(4) : ng\$(4)
PARA i = 1 A 17
INGRSAR Nombre del centro : nc\$(i)
PROCEDER
PROXIMO
ng\$(i) = "Cola"

Funciones Computadas

```
rg$(2) = 'Lima limón'
rg$(3) = 'Naranja'
rg$(4) = 'Pomelo'
RETOBNAR
```

Se trabaja nuevamente con dos matrices para calcular los costos $costo(1/7,4)$, y otra para almacenar la cantidad de los $cant(1/7,4)$. Ambas serán de 17 filas y 4 columnas.

Los valores a utilizar son 9

```
nc$(1/7) nombre de cada centro
nrocl(1/7) número de centro
cc(1/7) costo por centro
fc(1/7) facturación por centro
maxc(1/7) gaseosa de máximo envasado por centro
ccc(1/7) cantidad envasada por centro
ng$(4) nombre de cada gaseosa
ccgl(4) cantidad envasada por gaseosa
maxg(4) centro de mayor envasado para cada gaseosa
por(1/7) porcentaje que representa el costo de envasado
```

```
CARGA
PARA i = 1 A 7
  INGRESAR "Número de centro ", cc
  PARA j = 1 A 4
    INGRESAR "Número de gaseosa ", gas
    INGRESAR "Costo líquido ", c
    INGRESAR "Cantidad envasada ", ce
    cant(i,cen,gas) = ce
    costo(cen,gas) = (cl + 0.10 + 0.05) * ce
  SI cen = 5 OR cen = 10 OR cen = 16 ENTONCES
    costo(cen,gas) = costo(cen,gas) * 1.25
  FIN S
PROXIMO i
PROXIMO j
```

100. Algoritmos y estructuras de datos

Funciones Computadas

RETORNAR

Se cargan las dos matrices $cant(i,j)$ de declarar la cantidad envasada $cant(cen,gas)$ ce y la matriz $costo(i,j)$ con el valor que surge de sumar los costos de el líquido por litro, luego de envasarse y tener de obra por envasarse, y multiplicarlo por la cantidad envasada en $costo(cen,gas)$ $(cl + 0.10 + 0.05) * ce$. Además el costo se incrementa en 25%, $costo(cen,gas) = costo(cen,gas) * 1.25$, si los centros son el 5 o el 10 o el 16 por lo que se utiliza una OR para unir las condiciones $cen = 5$.

Si cen = 5 OR cen = 10 OR cen = 16.

CANTIDADCENTRO

```
PARA i = 1 A 17
  PARA j = 1 A 4
    cccl(i) = cc(i) + cant(i,j)
  PROXIMO j
PROXIMO i
RETORNAR
```

Se recorre a matriz por filas para acumular la cantidad de los envases por centro

CANTIDAD GASEOSA

```
PARA i = 1 A 4
  PARA j = 1 A 7
    ccgl(j) = ccgl(j) + cant(i,j)
  PROXIMO j
PROXIMO i
RETORNAR
```

Se recorre la matriz por columnas para acumular la cantidad de los envasados por gaseosa

```
COSTO CENTRO
PARA i = 1 A 17
  PARA j = 1 A 4
    cccl(j) = cccl(j) + costo(i,j)
  PROXIMO j
PROXIMO i
```

100. Algoritmos y estructuras de datos

EJERCICIOS COMBINADOS

REICRNAR

Se recorre la matriz por filas para acumular el costo de envasado por centro

MAXMO GASEOSA:

```
PARA i = 1 A 4
    max(i) = cc(i,1)
    PARA j = 2 A 7
        S cc(i,j) > max(i) ENTONCES max(i) = cc(i,j)
    PROXIMO j
RETORNAR
```

Se realiza max(i) con el valor envasado para esa gaseosa por el número centro max(i) = cc(i,1) y se compara con los 6 restantes. PARA i = 2 A 7

PORCENTAJE:

```
cost = 0
PARA i = 1 A 17
    cte = cost + cc(i)
PROXIMO i
PARA i = 2 A 17
    Por(i) = cc(i) / cte * 100
PROXIMO i
RETORNAR
```

Se calcula en este caso el porcentaje que representa el costo de envasado de cada centro, cc(i), sobre el costo total de la empresa, acumulador cte

```
PARA j = 1 A 17
    PARA i = 1 A 4
        tc(i) = cc(i) * 2
    PROXIMO i
PROXIMO j
RETORNAR
```

EJERCICIOS COMBINADOS

La facturación por centro se obtiene duplicando el costo por centro $cc(i) * 2$, ya que sabemos que el precio final por litro surge de recargar un 100% el costo

MAXIMO CENTRO:

```
PARA i = 2 A 17
    max(i) = 2 * cost(i,1)
    PARA j = 2 A 4
        S 2 * cost(i,j) > max(i) ENTONCES max(i) = 2 * cost(i,j)
    PROXIMO j
PROXIMO i
RETORNAR
```

Se determinó la máxima facturación de una a las 4 gaseosas para los 17 centros. Se supone que el máximo corresponde a la primera gaseosa max(i) = 2 * cost(i,1), y se compara con las 3 restantes. PARA j = 2 A 4. Si todavía haber utilizando una tercera matriz facturación, para dado que se genera solamente en este subprograma, se prefiere trabajar con el producto por 2 de $cc(i) + 1$.

ORDEN.

```
cost = 17
k = 1
HACER MIENTRAS k <= 0
    k = 0
    PARA i = 1 A cost - 1
        S tc(i) > tc(i+1) ENTONCES
            aux = tc(i)
            tc(i) = tc(i+1)
            tc(i+1) = aux
            aux = nrocc(i)
            nrocc(i) = nrocc(i+1)
            aux = cec(i)
            cec(i) = cec(i+1)
            cec(i+1) = aux
            aux = cc(i)
            cc(i) = cc(i+1)
            cc(i+1) = aux
```

INSTRUMENTOS
 PROXIMO
 con la
 RPTIR
 RITORNAR

Se ordenaron 4 vectores a función de la facturación de cada centro $fc(i)$

IMPRESION
 PARA 1 1 A 1/

MPRIMIR "Costo de envío", $cc(i)$, "Cantidad emvasada", $cc(i)$, "Litros",
 Costo de emvasado, $cc(i)$, "\$", "Facturado", $fc(i)$, "\$"
 PROXIMO

MPRIMIR "Cantidad emvasada por gascosos",
 PARA 1 1 A 4
 MPRIMIR "Gasosos", $ng(i)$, $cc(i)$, "Litros"

PROXIMO
 MPRIMIR "Centro de máxima cantidad en litros en gascosos",
 PARA 1 1 A 4

MPRIMIR "Gasosos", $ng(i)$
 PARA 1 1 A 7
 Si $cc(i) > maxj(j)$ ENTONCES IMPRIMIR $cc(i)$
 PROXIMO

MPRIMIR "Porcentaje del costo de emvasado de cada centro sobre el",
 PARA 1 1 A 1/

MPRIMIR "Centro", $cc(i)$
 MPRIMIR "Porcentaje", $por(i)$, "%"
 PROXIMO

MPRIMIR "Gasosos de mayor facturación por centro",
 PARA 1 1 A 1/

MPRIMIR "Centro", $cc(i)$
 PARA 1 1 A 4
 Si $ng(i) > maxj(j)$ ENTONCES IMPRIMIR $ng(i)$
 PROXIMO

PROXIMO
 RITORNAR

Ejercicios Propuestos

- Una empresa se dedica a almacenar trigo y posterior distribución de cereales en el interior de las Para ello cuenta con 20 depósitos en "Capitales", que en general se ubican en las inmediaciones de las estaciones de ferrocarril.

1. Maíz
2. Trigo
3. Cebada
4. Centeno

Mensualmente la oficina central, recibe un puntaje de cada depósito con los siguientes datos:

Número de depósito
 Existencias de maíz
 Existencias de trigo
 Existencias de cebada
 Existencias de centeno

Se pide calcular e imprimir:

- a) Cantidad total de kilos almacenados de cada cereal
- b) Cantidad de kilos almacenados en cada depósito
- c) Nombre del cereal que almacenó mayor cantidad de kilos, para cada depósito.
- d) Máxima cantidad de kilos almacenados de cada cereal
- e) Depósito de mayor recaudación
- f) Cantidad de depósitos que tienen almacenado más de 50000 kilos, para los 4 cereales

Ejercicios Combinados

g) Porcentaje de kilos de café cereal sobre el total de kilos almacenados y nombre del cereal de porcer a máxima

2. Una facultad p...vada con...a profesores para cubrir el dictado de sus materias. Los profesores trabajan de marzo a diciembre, dependiendo su sueldo de la cantidad de horas trabajadas

En enero y febrero, cobran un valor que consiste en la mitad del sueldo de octubre

La facultad lleva un registro de cada profesor, en una planilla personal con los siguientes datos

NÚMERO DE PROFESOR

NOMBRE	CANTIDAD DE HORAS	NÚMERO DE CATEGORÍA

Los valores de las horas dependen de la categoría y son los siguientes.

- 1 Profesor Titular 15\$
- 2 Profesor Asociado 12\$
- 3 Profesor Adjunto 10\$
- 4 Jefe de Trabajos Prácticos 8\$
- 5 Ayudante de Hora 6\$

Se pide calcular e imprimir:

- Sueldo anual de cada profesor
- Total de sueldos pagados cada mes
- Para cada profesor la cantidad de sueldos de año mayores al sueldo promedio de cada mes
- Máximo sueldo pagado en cada mes
- Porcentaje que representa cada sueldo anual sobre el total

PROGRAMACIÓN Y ESTILIZACIÓN DE ALGORITMOS

Ejercicios Combinados

f) Nombre del profesor que haya trabajado mayor cantidad de horas en diciembre

Los datos a, a se deben imprimir ordenados en forma ascendente de acuerdo al número

NÚMERO DE PROFESOR	NOMBRE	SUELDO ANUAL	PORCENTAJE

3. Un laboratorio de productos medicinales, elabora 15 remedios en base a 4 drogas. El valor de cada remedio depende de la droga que utiliza. Por cada remedio, el laboratorio tiene un registro para los 12 meses de año, con los siguientes datos

NÚMERO DE REMEDIO:
NOMBRE
DROGA UTILIZADA: (número)
CANTIDAD (en mg/gramos)
MES (número) CANTIDAD VENDIDA

Los valores de las drogas son los siguientes:

- | | |
|---------|----------|
| Droga 1 | 5\$ / mg |
| Droga 2 | 2\$ / mg |
| Droga 3 | 3\$ / mg |
| Droga 4 | 1\$ / mg |

El valor del remedio depende del valor de la droga usada y de la cantidad que se utilizó de la misma. Se deberá tener en cuenta que en los meses de marzo y de septiembre el precio de todos los remedios, sufrirá un descuento del 20%.

PROGRAMACIÓN Y ESTILIZACIÓN DE ALGORITMOS

Ejercicios Combinados

NÚMERO DE MODELO	CANTIDAD VENDIDA	TOTAL FACTURADO

7. Una empresa de cemento, tiene 12 centros de embolsado de sus materiales. En cada centro trabajan un promedio de 120 personas. Variamente se procede al embolsado, teniendo como mínimo que llenar 1000 bolsas y como máximo 2000. Los materiales que se embolsan son:

- 1 Ca
- 2 Arena
- 3 Cemento
- 4 Gravello

El costo de embolsado varía para cada centro y cada material. El costo de la hora de cada material es de 0.25\$. El costo de la mano de obra es de 0.15\$ por hora.

Para los centros 5, 6 y 12, hay que considerar que los costos se incrementan en un 25% debido a la ubicación alejada de dichos centros.

A costo de la bolsa, se le carga un 200% para calcular el precio de la bolsa.

Cada centro entrega al cabo de 20 días de trabajo una parilla con la cantidad de bolsas llenadas en ese lapso para cada material.

NÚMERO DE CENTRO		NOMBRE DEL CENTRO
NÚMERO MATERIAL	COSTO DE EMBOLSADO	CANTIDAD ENVASADA

Se pide:

- a) Cantidad embolsada por centro

Ejercicios Combinados

- b) Cantidad embolsada de cada material
- c) Costo de embolsado de cada centro
- d) Para cada material, nombre del centro en el que se embolsaron cuyo costo total de la empresa.
- e) Porcentaje que representa el costo de embolsado de cada centro sobre el costo total de la empresa.
- f) Si consideramos que se vende todo lo que se produce, ¿Cuál fue la facturación total de cada centro?
- g) Nombre del material de mayor facturación para los 12 centros

Se pide además imprimir los items a, c y f, ordenados de manera ascendente por facturación

Apéndice

Resumen de Instrucciones

INSTRUCCIONES DE ENTRADA

Sintaxis `INGRESAR variable`

Sintaxis `INGRESAR variable1, variable2, ..., variableN`

Sintaxis `INGRESAR {comentario}, variable`

INSTRUCCIONES DE SALIDA

Sintaxis `IMPRIMIR "comentario"`

Sintaxis `IMPRIMIR variable`

Sintaxis `IMPRIMIR {variable1, variable2, ..., variableN}`

Sintaxis `IMPRIMIR constante`

Sintaxis `IMPRIMIR expresión`

Sintaxis `IMPRIMIR "comentario", variable`

INSTRUCCIONES DE ASIGNACIÓN

Sintaxis: `variable = constante`

Sintaxis: `variable = expresión`

Sintaxis: `variable = variable2`

INSTRUCCIONES DE DECISION

- Instrucción de decisión simple `Si ... ENTONCES`

Sintaxis: `Si condición ENTONCES instrucción`

- Instrucción de decisión doble `Si ... ENTONCES ... SINO ...`

Sintaxis: `Si condición ENTONCES instrucción1 SINO instrucción2`

- Instrucción de decisión en bloques

Sintaxis: `Si condición ENTONCES`

`SINO`
`instrucción(s)`

`instrucción(s)`

`FIN SI`

Sintaxis: `Si condición ENTONCES instrucción1 SINO instrucción2`

`instrucción(s)`

`instrucción(s)`

`FIN SI`

- Instrucción de decisión múltiple `SELECCIONAR CASO`

210. Algoritmos y estructuras de datos

Sintaxis:

`SELECCIONAR CASO variable`
`CASO condición1`

`instrucción(s)`

`CASO condición2`

`instrucción(s)`

`CASO condición3`

`instrucción(s)`

`OTRO CASO`

`instrucción(s)`

`FIN SELECCIONAR`

INSTRUCCIONES DE REPETICIÓN

- Instrucción HACER MIENTRAS

Sintaxis: `HACER MIENTRAS condición`

`instrucción(s)`

`REPTIR`

- Instrucción HACER HASTA

Sintaxis: `HACER`

`instrucción(s)`

`REPETIR HASTA condición`

- Instrucciones PARA PRÓXIMO

Sintaxis: `PARA variable índice .. valor inicial A valor final PASO incremento`

`instrucción(s)`

`PROXIMO`

Algoritmos y estructuras de datos

Bibliografía

- Alcalde, Eduardo y García, Miguel: "Metodología de la programación", McGraw-Hill, 1992.
- Brassard, G y Bratley, P: "Algorithmitique", Masson, 1987.
- Braunstein, Silvia y Gioia, Alicia: "Introducción a la programación y a las estructuras de datos, Eudeba, 1986.
- Clavel, Fiordi: "Algorítmica y lenguajes", Masson, 1985.
- Clavel, Biondi: "Estructuras de datos", Masson, 1985.
- Coleman, D: "Organización de datos y programación estructurada, Gili, 1986.
- Courlin, y Kowarski, E: "Introducción a la programación y a las estructuras de datos", Dunod, 1987.
- Dijkstra, E: "A discipline of programming", Prentice Hall, 1976.
- Guilbur, R: "Procedimientos de clasificación", Masson, 1987.
- Harel, David: "Algorithimics", Addison Wesley, 1987.
- Joyanes, Luis: "Fundamentos de programación", McGraw-Hill, 1996.
- Lewis, J y Smith, M: "Estructuras de datos", Paraninfo, 1985.
- Lipschutz, Seymour: "Estructura de datos", McGraw-Hill, 1986.
- Rodriguez Almeida, Miguel Angel: "Metodología de la programación", McGraw-Hill, 1993.

Sabbadini, Domenico: "Introduzione alla programmazione strutturata", Bulci Editore, 1985.

Wirth, Niklaus: "Introducción a la programación sistemática", El Ateneo, 1984.

Wirth, Niklaus: "Algoritmos y Estructuras de datos", Prentice Hall, 1987.

Se terminó de imprimir en quince estudio gráfico
Agustín de la Vega 1555 • B1683BXC Martín Coronado • 49734359
en Abril del 2000

